ATTITUDE MODEL OF A REACTION WHEEL/ FIXED THRUSTER BASED SATELLITE USING TELEMETRY DATA

THESIS

Jason E. Smith
Captain, USAF

AFIT/GA/ENY/05-M010

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GA/ENY/05-M010

ATTITUDE MODEL OF A REACTION WHEEL/FIXED THRUSTER BASED SATELLITE USING TELEMETRY DATA

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Jason E. Smith, B.S.

Captain, USAF

March 2005

AFIT/GA/ENY/05-M010

ATTITUDE MODEL OF A REACTION WHEEL/ FIXED THRUSTER BASED SATELLITE USING TELEMETRY DATA

THESIS

Jason E. Smith, B.S.

Captain, USAF

Approved:

\\signed\\	09 Mar 05
Dr. Richard G. Cobb Thesis Advisor	Date
\\signed\\	09 Mar 05
Dr. Bradley S. Liebst Committee Member	Date
\\signed\\	09 Mar 05
LtCol Nate Titus Committee Member	Date

Acknowledgements

First and foremost, I am forever in debt to a special group of video game programmers for the their hours spent in teaching me a simple concept: quaternions.

I also thank Mr. Wilbur Lacy for his help in fixing SimSat and for helping me install the Fiber-Optic Gyroscope.

Lastly, I would like to thank my beautiful wife. Without her love, understanding, support, and encouragement boosting my exponentially fading will-power, this paper would have never been completed.

Jason E. Smith

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abstract

Attitude determination of satellites is normally the job of inertial instruments, such as gyroscopes, or through sensing instruments, such as star trackers or Global Positioning Satellites (GPS). Satellite health monitoring systems watch and determine if the satellite deviates from its normal operating attitude orientation. Knowing the orientation of a satellite is essential in being able to control it in order to complete the satellite's designated mission. While there are a multitude of ways to determine a satellite's orientation, very little research has been done on determining if the attitude of a satellite can be determined directly from telemetry data of the attitude control systems and an accurate spacecraft model. The fidelity of a satellite attitude determination model required to get reasonable predictions from using only telemetry data of the attitude controllers, such as thruster on/off indicators and reaction wheel rotor speeds, is investigated. Experimental tests using telemetry data received from the Air Force Institute of Technology's (AFIT) Simulated Satellite, SimSat, is used in verifying a Matlab® model which outputs SimSat's orientation from SimSat's reaction wheel and thruster telemetry data. Software modeling results showed that it is possible to determine a satellite's attitude from only the attitude controllers' telemetry data when the satellite's dynamic model is known. Testing involving SimSat showed that attitude determination from the Matlab® model is possible but not perfect. Additional information needs to be known about the satellite's systems and characteristics and about the environment in which the satellite operates, in order to increase the fidelity of the model for more accurate predictions of the satellite's attitude. Even though more research is needed, there is promise for using satellite attitude controllers for attitude determination in fields such as health monitoring and modeling and simulations.

ATTITUDE MODEL OF A REACTION WHEEL/ FIXED THRUSTER BASED SATELLITE USING TELEMETRY DATA

I. Introduction

As technology advances and spacecraft components get smaller, those interested in putting platforms into space are looking to maximize profits by including as many health monitoring sensors and redundancy systems as possible in order to keep operating time up and to extend the life span of the spacecraft as long as possible. The designs of those sensors and redundancy systems are based upon existing models of space and the spacecraft and how attitude controllers and determination devices actually react and perform in space. Without knowing its current attitude, the spacecraft cannot continue to meet its requirements, even if ways exist to control the spacecraft.

For example, what if a spacecraft in a certain known condition gets hit by debris or malfunctions, rearranging the configuration of the satellite without harming any critical systems during a blackout period with the ground-station? Let us assume that since the spacecraft cannot be seen and that there are no indications by onboard sensors that there was a change in configuration, such as bent solar panels which would slightly change the moment of inertia(MOI), and hence the dynamic model. On the next pass through the window, health monitoring software will pick-up that the spacecraft is out of alignment from the telemetry data being sent from the spacecraft. Assuming the spacecraft functionally checks out, the operators will then reposition the satellite. However, it is then only a matter of time before the satellite is out of alignment again due to the control laws using the original configuration parameters (based on a now incorrect dynamic model) causing expensive delays in

having to troubleshoot the problem. What can be done, using current telemetry data, to troubleshoot what went wrong and to correctly identify the current configuration parameters? Can we easily identify the model to get the satellite operational again?

Large sums of money are being spent on operational simulators that also use models of the space environment to test new codes and procedures before uploading new commands and programs on the actual platform to reduce the risk of permanently rendering the platform unusable. Operational simulators are also extremely useful in troubleshooting what went wrong or why a spacecraft may not be responding. In cases such as the Mars Rover, and in other troubleshooting related events of satellites in space, telemetry data is downloaded and used to try to recreate everything from some time before the event happened through until some time after in order to figure out what went wrong. Accurate models of the platform and its environment are needed and used in conjunction with the telemetry data for the creation of simulations. These simulations then try to give an approximate visual account of what was going on so that theories of what happened can be brought together and possible fixes analyzed [14].

1.1 Current Attitude Determination and Control Methods

In order to explore what can be done in troubleshooting spacecraft health problems and to simulate and model those problems so that a fix can be implemented, current capabilities in spacecraft attitude determination and control methods must be introduced and breifly explained. NASA [8] defines spacecraft attitude determination as

> the pointing direction of the orbiting satellite with respect to known objects; that is, the sun, moon, earth, stars, or earth's magnetic field direction. Attitude determination is the process of computing a set of parameters that describe this orientation. These parameters are computed from data that is downlinked (telemetry) to the tracking stations from the satellite. Attitude determination also includes evaluating the telemetry from the various onboard attitude sensors for any sign of phys-

ical deterioration, improper configuration, or changes in calibration or alignment.

The accuracy requirement for attitude determination is mission dependent. Some satellites require only that their sensors stay pointed towards the earth, while others require higher accuracy in order to observe a particular spot on the earth. Current attitude determination methods and their respective accuracies are summarized in Table 1.1 [17]. A brief summary of the determination methods are given below.

Table 1.1: Current Attitude Determination Methods

Sensor	Accuracy	Power (W)	Mass (kg)
Horizon Sensor	$0.1 - 1^\circ$	$0.3 - 5$	$0.5 - 3.5$
Sun Sensor	$0.005 - 3.0^\circ$	$0 - 3$	$0.04 - 2.0$
Global Positioning System Receivers (GPS)	N/A	N/A	N/A
Magnetometers	$0.5 - 3^\circ$	< 1	$0.3 - 1.2$
Star-Trackers (Star Sensors)	$0.0003 - 0.012^\circ$	$5 - 20$	N/A
Gyroscopes	1° /hr	$10 - 200$	$1 - 15$

Horizon sensors use the Earth's horizon to determine spacecraft attitude. Sun sensors use the Sun to determine spacecraft attitude and are currently the attitude determination device most commonly used. Global Positioning System (GPS) uses a spread-spectrum broadcast communication message that is exploited using relatively low-cost receivers in triangulating position based upon its orientation relative to the GPS satellites. Magnetometers can determine the attitude measured relative to the Earth's local magnetic field. Their accuracy is not as good as that of star or horizon sensors. However, these lower accuracies are far exceeded by the simplicity, reliability, lightweight, and low cost of magnetometers. Star-Trackers (Star Sensors) use observed star formations and compare them to a database of star formation information to determine the attitude of a spacecraft. These sensors allow the attitude to be measured extremely accurately. Most star sensors however are too slow to control a spacecraft's attitude directly. To address this slow processing, star sensors are normally complemented with gyroscopes for high accuracy and rapid response.

Gyroscopes may be used to measure the speed or angle of rotation of a spacecraft without any input from an external, absolute reference. They are designed with many different technologies: including spinning wheels, ring lasers, hemispherical resonating surfaces, and laser fiber optic bundles. Individual gyroscopes provide one or two axes of information, so multiple gyroscopes are often combined to form the Inertial Reference Unit (IRU) with three axes of information. IRUs combined with accelerometers are capable of sensing position and velocity. This setup is referred to as an Inertial Measurement Unit (IMU). Gyroscopes have the tendency to drift and thus need another instrument to re-calibrate themselves, such as star sensors.

Just knowing the current orientation to a high level of accuracy is not enough. Being able to maneuver and keep the satellite pointing in the desirable direction is just as important. Therefore, attitude determination and control work together to meet mission requirements. Current attitude control methods are summarized in Table 1.2 [17]. A brief summary of the control methods examined are given below.

Table 1.2: Current Attitude Control Methods

Control	Power (W)	Mass (kg)
Control Moment Gyros (CMG)	90	> 10
Gravity Gradient	0	0
Magnetic Torquer	$0.25 - 9.2$	$0.3 - 8.5$
Reaction Wheel	$1 - 10$	$0.3 - 3$
Cold Gas Thruster	$1.2 - 6$	$0.08 - 0.15 + fuel$

Control-moment gyros (CMG) consist of single- or double-gimbaled wheels spinning at a constant speed, and can produce large torque, depending on the angular velocity of the wheels and the rate of rotation of the gimbal. Gravity gradient stabilization is a passive attitude control technique that is designed to use the inertial properties of a vehicle to keep it pointed toward the Earth. Magnetic torquers are coils of uniform wire placed along an axis of rotation of the spacecraft. When a voltage is applied across a coil winding, a current is created, which creates a

linear magnetic dipole which interacts with the earth's magnetic field to produce torque. Magnetic torquers are used in numerous small spacecraft as well as larger and expensive spacecraft such as the Hubble Telescope.

Reaction wheels are flywheels attached to electric motors. When the motor applies a torque to speed up or slow down the flywheel, it produces a reacting torque on the body of the satellite. Over time, environmental torques such as atmospheric drag and solar pressure can cause a buildup of momentum in the reaction wheels. Without any means to dump this momentum, the reaction wheels would continually spin faster and faster until they reached maximum speed. A way to dump this extra momentum is to use magnetic torquers to produce an external torque in order to reduce the reaction wheel momentum.

Finally, cold gas thrusters are composed of a pressurized gas, a valve, and a convergent/divergent exit nozzle to provide low specific impulse in the conversion of the pressurized fuel to thrust. Instead of using cold pressurized gas, some systems may use a hot gas that is created from chemical reactions. In addition to cold and hot gas thrusters, there is research being done in pulsed plasma and ionized gas drives to replace cold gas thrusters due to the amount of fuel needed onboard to use cold gas jets.

There is a great amount of research and materials that can be found on the current methods of attitude determination and control. It is also obvious that unless the current orientation is known, being able to rotate and move a spacecraft becomes pointless. One question asked is whether or not it is possible to determine orientation and keep track of orientation from the attitude controller signals alone? As with current methods of attitude determination, a reference attitude still needs to be determined. But is it possible that given a reference orientation, real-time attitude determination at an acceptable accuracy can be found using just the information from the controllers themselves? Could an attitude determination system based

upon attitude controllers be used to increase accuracy if used in conjunction with other methods such as horizon detectors which can provide initial orientation?

1.2 Health Monitoring

If it were possible to determine orientation from the attitude controllers, one area that would benefit is satellite health and status monitoring. A satellite's health and status is monitored during each real-time pass. Local operating procedures developed by operations teams during pre-launch define key satellite housekeeping parameters that are verified during each pass. Software tools perform a majority of satellite telemetry monitoring. Data is sent through algorithms, and warning flags alert operators if something is wrong with the spacecraft. Also, the satellite's state-of-health and performance is monitored off-line through engineering and trend analysis telemetry processing much in the same way aircraft onboard flight computer data is downloaded and reviewed off-line to predict and prevent upcoming failures based upon data trends.

When the satellite is determined to be in an unexpected configuration, or established operating limits are violated, an anomaly investigation ensues. An anomaly report is filed and additional personnel are notified so that troubleshooting can begin. If the anomaly is determined to have a pre-approved response, and enough time remains in the current window, it is immediately implemented by the satellite operators. If it is not a pre-approved anomaly, then a plan of resolution is put together, approved, and implemented.

Research on health monitoring of spacecraft has been on-going since commercialization of space began. Extra instruments, which measure vibrations, are added to spacecraft in attempts to capture trend data for predicting pending failures of onboard equipment, such as reaction wheels [5]. Other systems, such as the Space Shuttle or the International Space Station, are mission critical systems that need to be monitored around the clock because failure of a system could mean loss of life [4].

An example of current space health monitoring being researched today is from a paper published in 2004 by R. W. Johnson and S. Jayaram [9]. This paper explores

> a new real-time detection/diagnosis methodology for an automated ground health monitoring system which are focused on the identification of abnormal transient response profiles from a satellite 6-DOF attitude control platform.

Research continues in determining the best algorithms needed in order to catch a problem from the existing telemetry data that is received. New algorithms are generated and researched when someone thinks of a new way of exploiting the data that already exists. Algorithms could be created that would compare the attitude determined from the controllers to other onboard attitude sensors. In March 2003, Capt Dabrowski investigated using certain controlled maneuvers to try to detect an uncooperative docking from estimated satellite moments of inertias [3]. What if instead of using his technique to detect uncooperative docking, it was modified to determine a change in configuration of the satellite based upon a change in moments of inertia? An algorithm like this could detect and be used to help determine the change in configuration of a satellite, and in turn be used to predict the resulting attitude from a series of manuevers.

1.3 Modeling and Simulation

In a paper about the future of Spacecraft Simulators given in 1998, Conrad Morris and Derek Rothwell summed up the vast uses of spacecraft simulators [13].

> Spacecraft simulators exist to : train operators; validate operational procedures (including innovative procedures that may be used for disaster recovery or to compensate for failing onboard instruments); validate onboard software patches and investigate anomalous behavior.

Spacecraft simulators are used for a plethora of reasons, all of which are in hopes of preventing loss of money due to a non-functioning spacecraft. In order for the Sim-

ulators to work as intended, realistic and accurate physics models of the spacecraft and environment need to be understood and created.

For example, let us assume that in a specific maneuver sequence telemetry data seems to indicate a sudden and large deviation from its expected course. An algorithm is triggered and a health monitoring system notifies the operator that there is an anomaly: troubleshooting begins. One way to help troubleshoot a problem is to take the telemetry data collected and then attempt to recreate what happened in a visual model. The visual model can be a computer generated model or a physical based model like Georgia Institute of Technology's Spacecraft in Figure 1.1.

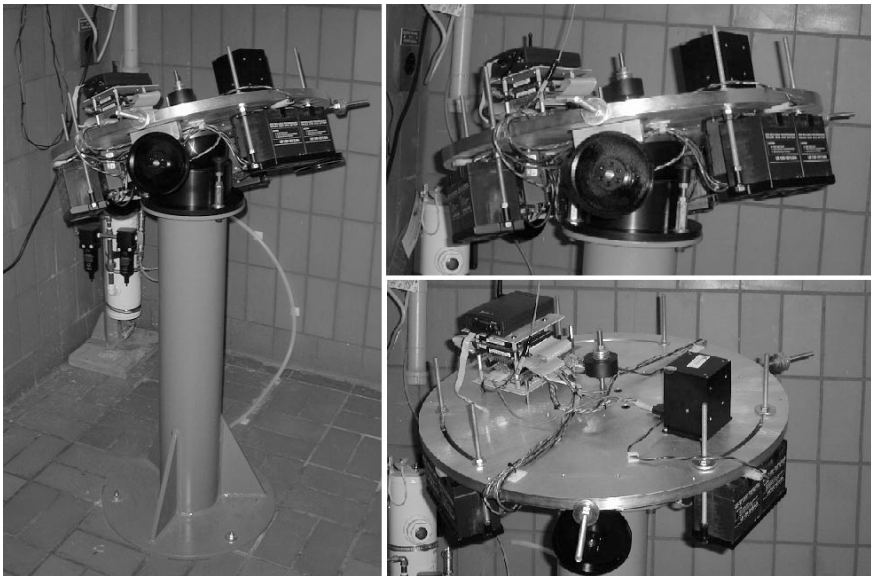

Figure 1.1: Georgia Institute of Technology's Spacecraft Simulator [10]

Research begins with trying to repeat the anomaly with the models so that a theory of what happened can be put together. In order to figure out what was occurring, accurate models of spacecraft and the environment will need to be used to determine the culprit. Also, once the problem is discovered, accurate models will need to be used to test the fix to ensure that it also works.

Accurate models of external force vectors acting on a satellite can be backed out using a combination of an attitude determination system and an attitude control

system. Instruments such as a gyroscope can give the actual attitude of the satellite based upon what happened. Control systems such as reaction wheels, which only apply internal torques to a system, can be used to model how the spacecraft should have maneuvered absent any external forces. The difference in how the spacecraft actually moved and how it should have moved with known forces would enable troubleshooters to figure out where an external force was acting that caused the anomaly seen in the data.

1.4 Research Objectives

There are plenty of potential uses for being able to use the attitude control system for attitude determination as presented in the previous sections. However, the first step is to determine if using an attitude control system to determine a spacecraft's attitude is feasible, and to gauge the accurracy required. Therefore, this objective of this thesis is to generate an attitude model of a reaction wheel/fixed thruster based satellite from telemetry data that is acquired from Air Force Institute of Technology's (AFIT) Simulation Satellite (SimSat), illustrated in Figure 1.2. This is in support of using SimSat as a model verification tool for a larger Matlab® based analysis tool.

There are two main criteria used in determining the success of this thesis:

- Achieving an accurate dynamic computer model in Simulink® using thrusters and reaction wheels as attitude controllers
- Having the computer model track SimSat from telemetry data received from just the attitude controller devices, i.e. the reaction wheels.

In addition to the two criteria, the existing mechanical gyroscopes on SimSat will be upgraded to new fiber-optic ones in support of future theses.

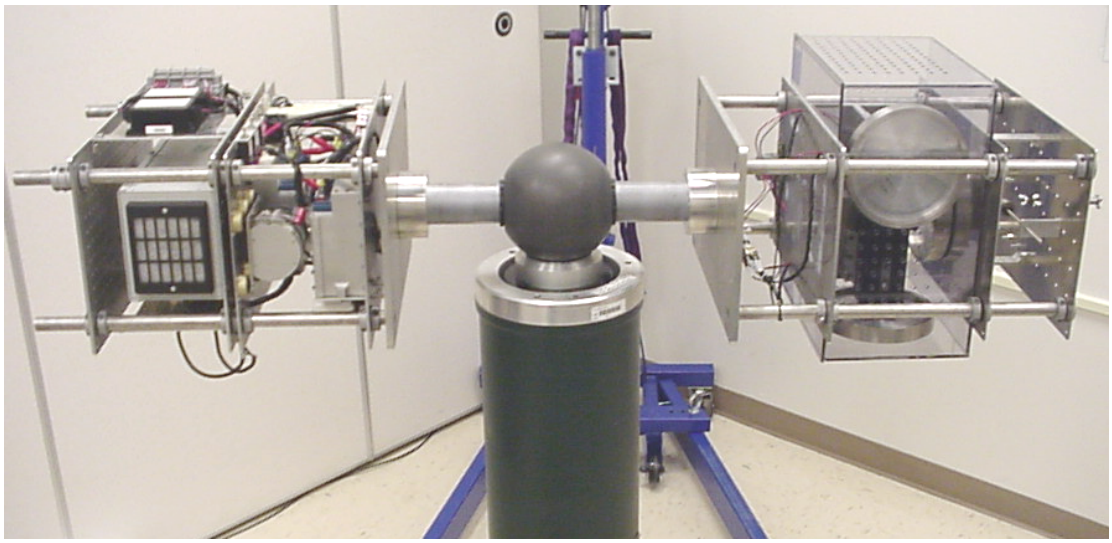

Figure 1.2: AFIT's SimSat

1.5 Thesis Outline

This chapter opened by posing a thought provoking question asking about some potential uses for determining attitude determination from telemetry data provided by the attitude controllers that can be found on spacecraft today. The focus of this research is most applicable to the areas of spacecraft health monitoring and spacecraft simulations, and has applications to autonomous spacecraft operations. It concluded with describing the objective for which this thesis is focused on. Chapter II develops the math and mechanics necessary in order to build an attitude determination model for a reaction wheel/fixed thruster based satellite. Chapter III describes the experimental test set-up. It covers both the simulation hardware, SimSat, and the software models which were used to simulate SimSat and the attitude determination model created. The results from the simulations conducted, and tests that were run are laid out in Chapter IV. Finally, Chapter V concludes with summarizing the main results and giving recommendations of where future research is needed.

II. Attitude Determination

From classical mechanics/dynamics, a rigid body has six degrees-of-freedom. Three degrees provide translational information. Three degrees provide rotational orientation information. SimSat is considered a rigid body which rests upon an air bearing assembly shown below.

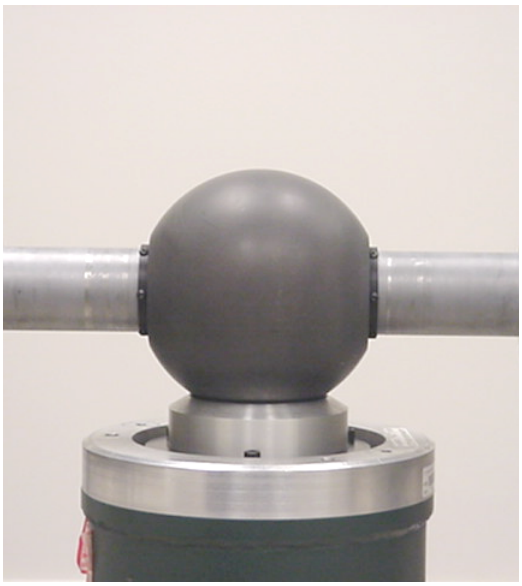

Figure 2.1: Space Electronics, Inc. Model SE9791 Tri-axis Spherical Air Bearing

SimSat's translational equations are absent due to the experimental set-up. Therefore, this thesis assumes that the translational information of the satellite is known (actual satellite translation can be determined through radar ground stations) and will thus concentrate on the three remaining degrees-of-freedom which describes the rotational characteristics of a rigid body.

Since SimSat is considered a rigid body for simulation purposes, this chapter begins with a quick overview of rigid body dynamics. Three different mathematical approaches to rotational kinematics will then be evaluated to determine which bests suits the need for meeting the objectives. This chapter wraps up with a section on

three axis control for satellite attitude control so that a model can be developed based upon the data retrieved from SimSat's reaction wheel data.

2.1 Rigid Body Dynamics

A rigid body is defined as a collection of particles that remain at fixed distances from each other at all times. Whereas particles are

> masses treated as if they were dimensionless points, [rigid bodies have] physical size and can thus rotate as well as translate... A rigid body is thus a dynamical system with, in general, six degrees of freedom. Three degrees of freedom are associated with the translational motion of some given point in the body, usually the center of mass, while three degrees of freedom describe the rotational motion of the system [18].

2.1.1 Translational Motion. The three component equations that describe the translational motion of a rigid body are the same equations used for motion of a mass particle which are found by applying Newton's Laws of Motion to each of the particles in the system.

$$\mathbf{F}_i = m_i \mathbf{a}_i \tag{2.1}$$

where $\mathbf{F_i}$ is the force acting on each particle with mass m_i and acceleration $\mathbf{a_i}$. Because of Newton's Third Law, the internal forces add up to zero leaving only the external forces acting on the body. Defining the center of mass, $\mathbf{r_{cm}}$, of a rigid body as the point in three dimensional space where the weighted average of the displacement of those small particles which make-up the rigid body equals zero

$$\mathbf{r}_{cm} = \frac{1}{M} \sum_{i=1}^{N} m_i \mathbf{r}_i \tag{2.2}$$

where M is the total mass of the body and $\mathbf{r_i}$ is the displacement of each particle from some reference point. The acceleration of the rigid body at the center of mass can then be found by differentiating $\mathbf{r_{cm}}$ twice, transforming Equation 2.1 into

$$\mathbf{F}_e = M\frac{\partial^2 \mathbf{r}_{cm}}{\partial t^2} \tag{2.3}$$

The center of mass behaves as a point where all the mass of the rigid body is concentrated and thus is where the total external force acts [18].

2.1.2 Rotational Motion.

> While the center of mass provides valuable information and simplifies the analysis of translational motion, it gives no measure of the way the mass is distributed on the body. The mass of a body describes the amount of matter contained in the body and the resistance of the body to translational motion... A quantity that describes the resistance of a body to rotation [is a quantity that] is dependent on how the mass is distributed. As the center of mass is located using the first moment of mass distribution... the second moment of the mass distribution [is considered in determining the mass distribution of the body.] [1]

Using the x-y-z coordinate system in Figure 2.2, there are two types of quantities that need to be defined to fully describe the mass distribution of a rigid body. The first is the distribution of mass with respect to an axis. The second is the distribution of mass with respect to a plane. Define the mass moment of inertia about an axis as how much mass is displaced from a certain axis. As is shown in Figure 2.2, the mass moment of inertia about the x-axis is

$$\mathbf{I}_{xx} = \int_{body} \mathbf{R}_x^2 dm = \int_{body} (y^2 + z^2) dm \tag{2.4}$$

Similarly, for the y-axis and z-axis respectively

$$\mathbf{I}_{yy} = \int_{body} \mathbf{R}_y^2 dm = \int_{body} (x^2 + z^2) dm \tag{2.5}$$

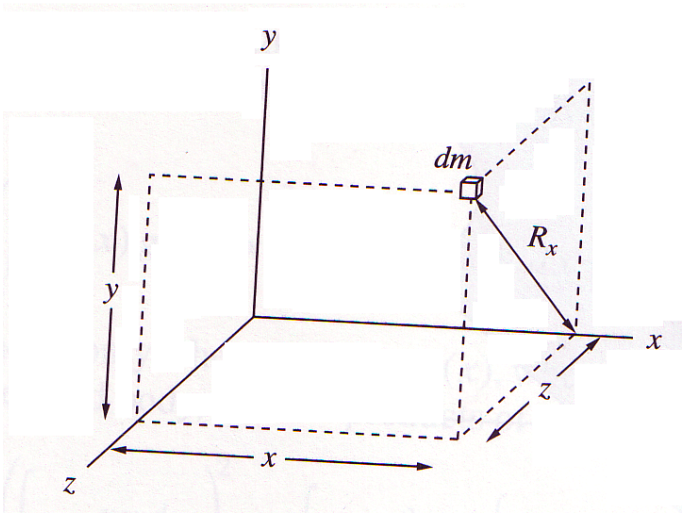

Figure 2.2: x-y-z Coordinate System of a Body [1]

$$\mathbf{I}_{zz} = \int_{body} \mathbf{R}_z^2 dm = \int_{body} (x^2 + y^2) dm \tag{2.6}$$

What these equations imply is that the more mass displaced away from the body axis, the body will have more resistance (inertia) to rotational motion about that axis.

The products of inertia are a measure of how much mass is displaced from a certain plane. The products of inertia are defined

$$\mathbf{I}_{xy} = \int_{body} xy dm = \mathbf{I}_{yx} \tag{2.7}$$

$$\mathbf{I}_{xz} = \int_{body} xz dm = \mathbf{I}_{zx} \tag{2.8}$$

$$\mathbf{I}_{yz} = \int_{body} yz dm = \mathbf{I}_{yz} \tag{2.9}$$

about the xy, xz, and yz planes respectively.

The products of inertia describe certain symmetrical properties of a rigid body with respect to the coordinate axes. If there is symmetry with respect to the yz plane, then I_{xy}=I_{xz}=0. If there is symmetry with respect to the xz plane, then I_{xy}=I_{yz}=0. If there is symmetry with respect to the xy plane, then I_{xz}=I_{yz}=0. Putting the moments and products of inertia together in a matrix gives

$$[\mathbf{I}] = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix} \tag{2.10}$$

> If a rigid body is symmetric about an axis then it must have symmetry about at least two planes. Thus for a body that has an axis of symmetry, all products of inertia vanish when one of the coordinate axes is along the symmetry axis. It should be noted that a body need not have planes or axes of symmetry for the products of inertia to vanish. A proper orientation of the [coordinate axes] leads to the same result...If the coordinate axes are selected such that the products of inertia vanish, the coordinate axes are referred to as principal axes and the corresponding mass moments of inertia are called principal moments of inertia [1].

The inertia matrix of the principal moments of inertia is denoted by

$$[\mathbf{I}] = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} \tag{2.11}$$

Since, by definition, the mass of a rigid body does not change over time, the inertia matrix only needs to be recalculated if there is a change in the mass distribution of the rigid body. When a physical object is added or subtracted from SimSat, the inertial properties will change and the inertia matrix will need to be recalculated. In order to simplify analysis, SimSat uses a body fixed reference frame that assumes

it is aligned with the principal axes, thus allowing the use of the principal moments of inertia where the body axes are defined as in the following picture

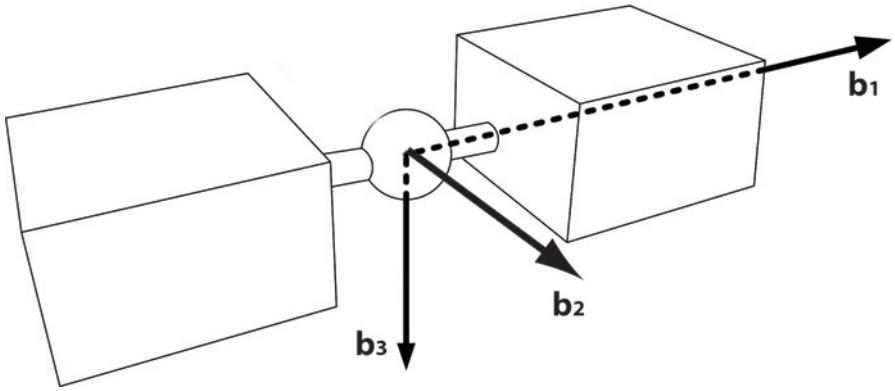

Figure 2.3: Right Handed Coordinate System

The angular momentum of a rigid body is directly proportional to how fast the rigid body is rotating, denoted by the angular velocity $\boldsymbol{\omega}$, and the resistance of the body to rotate, the inertia matrix, $\mathbf{I}$.

$$\overrightarrow{\mathbf{H}} = \mathbf{I}\overrightarrow{\boldsymbol{\omega}} \tag{2.12}$$

Taking the time derivative of the angular momentum, Equation 2.12, with respect to inertial space equals any applied torque to the rigid body. An origin needs to be chosen in order to calculate the moments. Since the center of mass is usually used in the translational equations to simplify calculations, it will also be advantageous to use the center of mass as the origin for the rotational equations. Using the center of mass as the origin of the principal/body axes, differentiating both sides of Equation 2.12 gives

$$\overrightarrow{\mathbf{M}} = \left(\frac{d\overrightarrow{\mathbf{H}}}{dt}\right)_{bodyframe} + \overrightarrow{\boldsymbol{\omega}} \times \overrightarrow{\mathbf{H}} \tag{2.13}$$

in the body frame. Applying Equation 2.11 and the matrix form of the cross product yields

$$\overrightarrow{\mathbf{M}} = \mathbf{I}\frac{d\overrightarrow{\boldsymbol{\omega}}}{dt} + \boldsymbol{\omega}^x \mathbf{I}\overrightarrow{\boldsymbol{\omega}} \tag{2.14}$$

where $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are vectors of body fixed angular velocities and accelerations respectively

$$\overrightarrow{\boldsymbol{\omega}} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.15}$$

$$\frac{d\overrightarrow{\boldsymbol{\omega}}}{dt} = \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} \tag{2.16}$$

and $\boldsymbol{\omega}^x$ is a skew-symmetric matrix of the form

$$\boldsymbol{\omega}^x = \begin{pmatrix} 0 & -\omega_3 & \omega_3 \\ \omega_3 & 0 & -\omega_2 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{2.17}$$

Substituting Equations 2.11, 2.15, 2.16, and 2.17 in Equation 2.14 gives

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{pmatrix} \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} + \begin{pmatrix} 0 & -\omega_3 & \omega_3 \\ \omega_3 & 0 & -\omega_2 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{pmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.18}$$

By selecting the body axes as principal axes, all products of inertia vanished and what is left is the widely known Euler's equations of motion for a rigid body

$$
\begin{aligned}
M_1 &= I_{11}\dot{\omega}_1 - (I_{22} - I_{33})\omega_2\omega_3 \\
M_2 &= I_{22}\dot{\omega}_2 - (I_{33} - I_{11})\omega_1\omega_3 \\
M_3 &= I_{33}\dot{\omega}_3 - (I_{11} - I_{22})\omega_1\omega_2
\end{aligned}
\tag{2.19}
$$

which is easily seen as a simplification of the more general form of the rotational equations in terms of the body-fixed angular velocities had the products of inertia not vanished [1].

$$
\begin{aligned}
M_{Gx} &= I_{xx}\dot{\omega}_x - I_{xy}(\dot{\omega}_y - \omega_x\omega_z) - I_{xz}(\dot{\omega}_z + \omega_x\omega_y) - (I_{yy} - I_{zz})\omega_y\omega_z - I_{yz}(\omega_y^2 - \omega_z^2) \\
M_{Gy} &= I_{yy}\dot{\omega}_y - I_{yz}(\dot{\omega}_z - \omega_x\omega_y) - I_{xy}(\dot{\omega}_x + \omega_y\omega_z) - (I_{zz} - I_{xx})\omega_x\omega_z - I_{xz}(\omega_z^2 - \omega_x^2) \\
M_{Gz} &= I_{zz}\dot{\omega}_z - I_{xz}(\dot{\omega}_x - \omega_y\omega_z) - I_{yz}(\dot{\omega}_y + \omega_x\omega_z) - (I_{xx} - I_{yy})\omega_x\omega_y - I_{xy}(\omega_x^2 - \omega_y^2)
\end{aligned}
\tag{2.20}
$$

2.2 Rotational Kinematics

In the previous section, three scalar equations, known as Euler's Equations, defining the rotational dynamics of a rigid body were found. However, those three equations are in reference to a body fixed frame. It is unlikely that observations from a body-fixed frame will provide the best point of view since most observations of a satellite, and specifically with SimSat, are from an inertial reference point that is found outside and away from the rigid body. A mathematical relationship is needed to link the body-fixed angular velocities and accelerations in the dynamics equations to the inertial orientation and orientation rates which are easily observed. This relationship is defined in rotational kinematics. There are three widely used methods in identifying rotational motion: Direction Cosines, Euler-Angles, and Quaternions.

2.2.1 Direction Cosines.

Given a unit vector $\mathbf{v}$, in an orthonormal coordinate frame $\{X, Y, Z\}$, the direction of cosines for $\mathbf{v}$ are the cosines of the angles α, β, and γ

between the vectors **X**, **Y**, and **Z** respectively, as is illustrated in Figure 2.4.

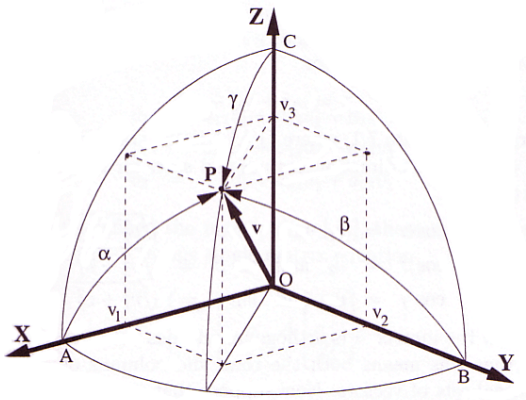

Figure 2.4: Direction Cosine Geometry [12]

If **v** and **X**, **Y**, and **Z** are unit vectors, then

$$\begin{aligned} \cos\alpha &= \mathbf{v} \bullet \mathbf{X} \\ \cos\beta &= \mathbf{v} \bullet \mathbf{Y} \\ \cos\gamma &= \mathbf{v} \bullet \mathbf{Z} \end{aligned} \tag{2.21}$$

Applying this to an orthonormal coordinate frame, $\{u, v, w\}$, a set of nine relationships is formed

$$\begin{matrix} \mathbf{u} \bullet \mathbf{X} & \mathbf{v} \bullet \mathbf{X} & \mathbf{w} \bullet \mathbf{X} \\ \mathbf{u} \bullet \mathbf{Y} & \mathbf{v} \bullet \mathbf{Y} & \mathbf{w} \bullet \mathbf{Y} \\ \mathbf{u} \bullet \mathbf{Z} & \mathbf{v} \bullet \mathbf{Z} & \mathbf{w} \bullet \mathbf{Z} \end{matrix} \tag{2.22}$$

where all the dot products are cosines of angles between two vectors. Putting these into a matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{u} \bullet \mathbf{X} & \mathbf{v} \bullet \mathbf{X} & \mathbf{w} \bullet \mathbf{X} \\ \mathbf{u} \bullet \mathbf{Y} & \mathbf{v} \bullet \mathbf{Y} & \mathbf{w} \bullet \mathbf{Y} \\ \mathbf{u} \bullet \mathbf{Z} & \mathbf{v} \bullet \mathbf{Z} & \mathbf{w} \bullet \mathbf{Z} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{2.23}$$

A becomes what is known as the Direction Cosine Matrix. Since the dot product is a description of a projection, **A** can be thought of as a projection of one orthonormal coordinate frame onto another, as a rotation matrix, or as a change of basis.

$$\begin{aligned} \mathbf{X} &= a_{11}\mathbf{u} + a_{12}\mathbf{v} + a_{13}\mathbf{w} \\ \mathbf{Y} &= a_{21}\mathbf{u} + a_{22}\mathbf{v} + a_{23}\mathbf{w} \\ \mathbf{Z} &= a_{31}\mathbf{u} + a_{32}\mathbf{v} + a_{33}\mathbf{w} \end{aligned} \tag{2.24}$$

In matrix form

$$\begin{Bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{Bmatrix} = \mathbf{A} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{Bmatrix} \tag{2.25}$$

Since $\{X, Y, Z\}$ and $\{u, v, w\}$ are each orthonormal coordinate frames

$$\mathbf{A}^{-1} = \mathbf{A}^T \tag{2.26}$$

which easily allows for a transformation back in the other direction by

$$\begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{Bmatrix} = \mathbf{A}^T \begin{Bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{Bmatrix} \tag{2.27}$$

Obviously at any specific point in time, the Direction Cosines are an easy way to convert from one reference frame to another. However, in dynamics, it is necessary to know how things behave over time. Looking at Equation 2.23 it can be seen that if the angles are not constant over time then the nine dot products of the direction cosine matrix are not constant. This means that in order to find all nine elements of the direction cosine matrix, **A**, nine integrations will be required. Since nine integrations at each time step can be mathematically costly and since direction cosines deal with the cosines of angles instead of the angles themselves, other alternatives need to be looked at.

2.2.2 Euler-Angles. It is a rarity not to discuss Euler-angles when reviewing rotations of coordinate frames in rotational kinematics. In the eighteenth century, Leonard Euler (1707-1778) proved a theorem which guarantees the existence of sequences of three rotations which relate two independent coordinate frames:

> Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis [12].

An Euler-angle is the angle of rotation about a coordinate axis. Three successive euler-angle rotations are known as an Euler-angle sequence. Due to the limitation that no two successive rotations may be about the same axis, i.e. two successive rotations can be summed into one, there are 12 possible combinations of Euler-angle sequences

$$\begin{array}{lll} xyz & yzx & zxy \\ xzy & yxz & zyx \\ xyx & yzy & zxz \\ xzx & yxy & zyz \end{array} \tag{2.28}$$

A popular sequence is a sequence known as the Aerospace Euler Sequence. It is also known as a 3-2-1 Euler-angle sequence or a zyx sequence since the first rotation

is about the 3- or z-axis followed by the 2- or y-axis and finishing the sequence with a rotation about the 1-or x-axis. Figure 2.5 illustrates this rotation sequence compared to an inertial frame $\{X, Y, Z\}$.

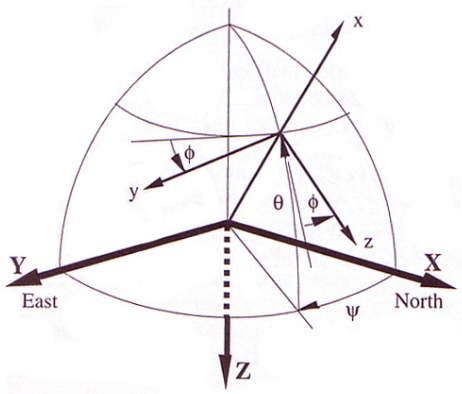

Figure 2.5: Aerospace Euler Sequence [12]

The rotations of the Aerospace Euler sequence is simply a set of rotations done starting from the reference or inertial frame and ending in the body frame. As can be derived from Figure 2.5, the three rotation matrices are:

$$\begin{aligned}
\mathbf{R}_{\psi} &= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{R}_{\theta} &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \\
\mathbf{R}_{\phi} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}
\end{aligned} \tag{2.29}$$

Putting the rotations together in sequence gives a matrix product from right to left

$$\mathbf{R} = \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi \tag{2.30}$$

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.31}$$

For SimSat, it is necessary to know what the Euler-angle angular rates are in terms of body-axis angular rates. Starting with

$$\mathbf{v}_b = \mathbf{R}\mathbf{v}_r \tag{2.32}$$

where $\mathbf{v}_r$ is a vector in the inertial frame, $\mathbf{v}_b$ is a vector in the body frame, and $\mathbf{R}$ is the rotational matrix from an Euler-angle sequence. Taking the derivatives with respect to time on both sides of the equation yields

$$\dot{\mathbf{v}}_b = \dot{\mathbf{R}}\mathbf{v}_r + \mathbf{R}\dot{\mathbf{v}}_r \tag{2.33}$$

Rearranging Equation 2.32 and substituting into Equation 2.33

$$\dot{\mathbf{v}}_b = \dot{\mathbf{R}}\mathbf{R}^{-1}\mathbf{v}_b + \mathbf{R}\dot{\mathbf{v}}_r \tag{2.34}$$

It is also known that the derivative of $\mathbf{v}_r$ is equal to the the derivative of $\mathbf{v_b}$ plus the cross product of the angular velocity of the body frame with respect to the inertial frame, $\boldsymbol{\omega}$, which all equals zero since the inertial frame is not rotating

$$\dot{\mathbf{v}}_r = \dot{\mathbf{v}}_b + \boldsymbol{\omega} \times \mathbf{v}_b = 0 \tag{2.35}$$

Solving for $\dot{\mathbf{v}}_b$ and substituting into Equation 2.34

$$\dot{\mathbf{v}}_b = \dot{\mathbf{R}}\mathbf{R}^{-1}\mathbf{v}_b = -\boldsymbol{\omega} \times \mathbf{v}_b = -\boldsymbol{\omega}^x \mathbf{v}_b \tag{2.36}$$

where $\boldsymbol{\omega}^x$ is the same skew-symmetric matrix found in Equation 2.17. Dividing through by $\mathbf{v}_b$ on each side and solving for the Euler-angle angular rates, $\dot{\mathbf{R}}$,

$$\dot{\mathbf{R}} = \mathbf{R}(-\boldsymbol{\omega}^x) \tag{2.37}$$

and after some tedious algebraic simplification [12], what is left is a result of the form

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.38}$$

From Equation 2.38, the $\tan\theta$ and $\sec\theta$ terms show that a singularity occurs when $\theta = 90°$. A system needs to be well-known so that a sequence can be chosen to take into consideration where the singularity occurs. Singularities for SimSat are only an issue when $\theta = 0°$ since the initial condition for θ is set to zero, where θ is measured from the horizontal axis. The air bearing assembly, Figure 2.1, limits SimSat's motion to $\pm 90°$ of pitch from the horizontal axis. This Aerospace Euler Sequence is acceptable and is in use in a previous thesis done with Simsat by Capt French [6]. However, this sequence is not the only one that can be used. Capt Fulton uses a 1-3-2 or xyz rotation sequence in his thesis and notes that a 1-2-3 rotation sequence has a singularity at $\theta = 0°$ and cannot be used [7].

Even though Euler-angles are more intuitive and require fewer integrations than Direction Cosines, there are more intensive trigonometric calculations and no unique set of rotations to represent an orientation. That, along with the inclusion of singularities, makes comparisons of different models that use different rotation sequences difficult.

2.2.3 Quaternions. Even though SimSat has physical limitations to avoid singularities, a satellite in outer-space does not. There is a mathematical approach, that uses less integrations than direction cosines, that has quicker computations, and eliminates the singularities of Euler-angles; this approach involves using quaternions. This is the preferred method used in modern spacecraft, graphics intensive computer games, and computer intensive simulations [12]. A brief overview of quaternions follow below.

In 1843 William Rowan Hamilton invented the quaternion, a hyper-complex number of rank 4. A central rule which quaternions are based upon is

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{2.39}$$

A quaternion is specified by the quantity

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{2.40}$$

where w,x,y, and z are real numbers. This quantity can be thought of as a vector $(w, x, y, z) \in R^4$. Quaternions that are related to rotations are unit length which are located on a hyper-sphere of radius 1, giving

$$w^2 + x^2 + y^2 + z^2 = 1 \tag{2.41}$$

Another way of looking at quaternions is to break $(w, x, y, z) \in R^4$ into a scalar part, denoted by w, and a vector part, $(x, y, z) \in R^3$ with $\mathbf{i}, \mathbf{j}, \mathbf{k}$ being an orthonormal basis so that

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = q_0 + \mathbf{q} \tag{2.42}$$

where q_0 is the scalar w and $\mathbf{q}$ is the vector $(x, y, z) \in R^3$.

From linear algebra, a set of quaternions, under the operations of addition and multiplication, form a non-commutative division ring. This means that quaternions are a field that is closed under addition and scalar multiplication, but the commutative law for multiplication does not hold for quaternion multiplication. Treating the quaternion like a vector in R^4 addition, equality and scalar multiplication of quaternions are as expected.

Let p and q be quaternions defined as

$$\begin{aligned} p &= p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k} = (p_0, p_1, p_2, p_3) = p_0 + \mathbf{p} \\ q &= q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} = (q_0, q_1, q_2, q_3) = q_0 + \mathbf{q} \end{aligned} \tag{2.43}$$

then $p = q$ if and only if

$$p_0 = q_0, p_1 = q_1, p_2 = q_2, p_3 = q_3 \tag{2.44}$$

and $p + q$ is

$$p + q = (p_0 + q_0, p_1 + q_1, p_2 + q_2, p_3 + q_3) \tag{2.45}$$

and let c be any scalar such that $c \in R$ so that

$$cp = (cp_0, cp_1, cp_2, cp_3) \tag{2.46}$$

$\mathbf{i}, \mathbf{j}, \mathbf{k}$ are an orthonormal basis that must follow the special products of Equation 2.39, leaving the following relationships, i.e. the right hand rule

$$\begin{aligned} \mathbf{ij} &= \mathbf{k} = -\mathbf{jk} \\ \mathbf{jk} &= \mathbf{i} = -\mathbf{kj} \\ \mathbf{ki} &= \mathbf{j} = -\mathbf{ik} \end{aligned} \tag{2.47}$$

With this in mind, quaternion multiplication can be defined as

$$\begin{aligned} pq &= (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ &= p_0q_0 + p_0q_1\mathbf{i} + p_0q_2\mathbf{j} + p_0q_3\mathbf{k} \\ &+ p_1q_0\mathbf{i} + p_1q_1\mathbf{i}^2 + p_1q_2\mathbf{ij} + p_1q_3\mathbf{ik} \\ &+ p_2q_0\mathbf{j} + p_2q_1\mathbf{ji} + p_2q_2\mathbf{j}^2 + p_2q_3\mathbf{jk} \\ &+ p_3q_0\mathbf{k} + p_3q_1\mathbf{ki} + p_3q_2\mathbf{kj} + p_3q_3\mathbf{k}^2 \end{aligned} \tag{2.48}$$

Which, when simplified, equates to

$$\begin{aligned} pq \quad = \quad & p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) \\ & + p_0(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + q_0(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \\ & (p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} + (p_1q_2 - p_2q_1)\mathbf{k} \end{aligned} \tag{2.49}$$

This can also be written in a vector equation consisting of dot and cross products

$$pq = p_0q_0 - \mathbf{p} \bullet \mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q} \tag{2.50}$$

As complex numbers have a conjugate, quaternions, being hyper-complex, also have a conjugate which can be defined as: if $q = q_0 + \mathbf{q}$ then

$$q^* = q_0 - \mathbf{q} \tag{2.51}$$

which makes the sum of a quaternion and its conjugate

$$q + q^* = 2q_0 \tag{2.52}$$

The norm of a quaternion is defined as

$$N(q) = \sqrt{q^* q} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{2.53}$$

And the inverse of a unit quaternion is

$$q^{-1} = q^* \tag{2.54}$$

Above, the algebra of quaternions have been defined, but it does not show exactly how quaternions are related to rotations. A unit quaternion, $|q| = 1$, can be written as

$$q = q_0 + \mathbf{q} = \cos\theta + \mathbf{u}\sin\theta \tag{2.55}$$

where θ is an angle and $\mathbf{u} = \mathbf{q}/\mathbf{q_0}$ is a unit vector. There is a theorem that states

> For any unit quaternion, [Equation 2.55], and for any vector $\mathbf{v} \in R^3$ the action of the linear operator
>
> $$\mathrm{L}_q(\mathbf{v}) = q\mathbf{v}q^* \tag{2.56}$$
>
> on $\mathbf{v}$ may be interpreted geometrically as a rotation of the vector $\mathbf{v}$ through an angle 2θ about $\mathbf{q}$ as the axis of rotation [12].

Figure 2.6 below is a physical representation of a quaternion rotation. The quaternion, Equation 2.55, can be thought of as a rotation in R^3 through an angle 2θ about $\mathbf{q}$ as its axis. There is another theorem which states

$$\mathrm{L}_{q^*}(\mathbf{v}) = q^* \mathbf{v} q \tag{2.57}$$

> may be geometrically interpreted as a rotation of the coordinate frame with respect to the vector $\mathbf{v}$ through angle 2θ about $\mathbf{q}$ axis or an opposite rotation of vector $\mathbf{v}$ with respect to the coordinate frame through an angle 2θ about $\mathbf{q}$ as the axis [12].

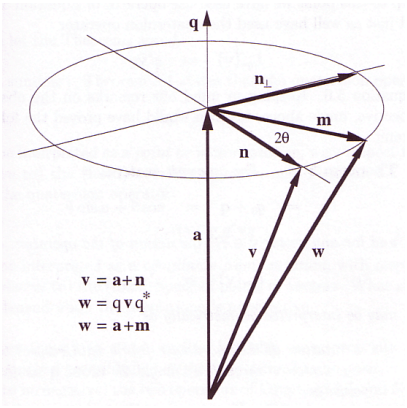

Figure 2.6: Rotation Operator Geometry [12]

Equation 2.56 may be interpreted as a point or vector rotation with respect to a fixed reference frame and Equation 2.57 as a coordinate frame rotation relative to a fixed vector or point in space. Just as in Direction Cosines and Euler-angle rotation sequences, quaternion rotation operators can be multiplied together to get a rotational sequence as stated in this next theorem:

> Suppose that p and q are unit quaternions which define the quaternion rotation operators

$$\mathrm{L}_p(\mathbf{u}) = p\mathbf{u}p^* \quad and \quad \mathrm{L}_q(\mathbf{v}) = q\mathbf{v}q^* \tag{2.58}$$

Then the quaternion product qp defines a quaternion rotation operator L_{qp} which represents a sequence of operators, L_p followed by L_q. The axis and the angle of rotation are those represented by the quaternion product, say, $r = qp$ [12].

This last theorem allows a transformation of Euler-angle sequences to quaternions to be derived. Taking the Aerospace Euler Sequence found in the previous section, let

$$\begin{aligned} q_z &= \cos\tfrac{\psi}{2} + \mathbf{k}\sin\tfrac{\psi}{2} \\ q_y &= \cos\tfrac{\theta}{2} + \mathbf{j}\sin\tfrac{\theta}{2} \\ q_x &= \cos\tfrac{\phi}{2} + \mathbf{i}\sin\tfrac{\phi}{2} \end{aligned} \tag{2.59}$$

Then defining q as the rotation product

$$q = q_z q_y q_x = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{2.60}$$

where

$$\begin{aligned} q_0 &= \cos\tfrac{\psi}{2}\cos\tfrac{\theta}{2}\cos\tfrac{\phi}{2} + \sin\tfrac{\psi}{2}\sin\tfrac{\theta}{2}\sin\tfrac{\phi}{2} \\ q_1 &= \cos\tfrac{\psi}{2}\cos\tfrac{\theta}{2}\sin\tfrac{\phi}{2} - \sin\tfrac{\psi}{2}\sin\tfrac{\theta}{2}\cos\tfrac{\phi}{2} \\ q_2 &= \cos\tfrac{\psi}{2}\sin\tfrac{\theta}{2}\cos\tfrac{\phi}{2} + \sin\tfrac{\psi}{2}\cos\tfrac{\theta}{2}\sin\tfrac{\phi}{2} \\ q_3 &= \sin\tfrac{\psi}{2}\cos\tfrac{\theta}{2}\cos\tfrac{\phi}{2} - \cos\tfrac{\psi}{2}\sin\tfrac{\theta}{2}\sin\tfrac{\phi}{2} \end{aligned} \tag{2.61}$$

To go from quaternions to Direction Cosines, set the individual elements of matrix $\mathbf{R}$ that is found in Equation 2.31 to the individual elements in the following matrix

$$\mathbf{R} = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix} \tag{2.62}$$

From this it is easy to go from quaternions back to euler-angles using the Direction Cosine matrix in Equation 2.31 and the equivalent matrix in Equation 2.62.

$$\begin{aligned} \tan\psi &= \tfrac{m_{12}}{m_{11}} \\ \sin\theta &= -m_{13} \\ \tan\phi &= \tfrac{m_{23}}{m_{33}} \end{aligned} \tag{2.63}$$

where

$$\begin{aligned} m_{11} &= 2q_0^2 - 1 + 2q_1^2 \\ m_{12} &= 2q_1q_2 + 2q_0q_3 \\ m_{13} &= 2q_1q_3 - 2q_0q_2 \\ m_{23} &= 2q_2q_3 + 2q_0q_1 \\ m_{33} &= 2q_0^2 - 1 + 2q_3^2 \end{aligned} \tag{2.64}$$

In order to find the derivative, a transition quaternion, Δr, must be used to relate q(t) and $q(t + \Delta t)$

$$q(t + \Delta t) = q(t)\Delta r(t) \tag{2.65}$$

where

$$\Delta r(t) = \cos\left(\frac{\Delta\alpha}{2}\right) + \mathbf{v}(t)\sin\left(\frac{\Delta\alpha}{2}\right) \tag{2.66}$$

Since Δt can be chosen, it will be assumed chosen small enough in order to apply small angle approximations to the $\Delta\alpha$

$$\Delta r(t) = 1 + \mathbf{v}(t)\frac{\Delta\alpha}{2} \tag{2.67}$$

then

$$q(t+\Delta t) = q(t)\left[1 + \mathbf{v}(t)\frac{\Delta\alpha}{2}\right] \tag{2.68}$$

After some rearranging and dividing both sides by Δt

$$\frac{q(t+\Delta t) - q(t)}{\Delta t} = \frac{1}{2}q(t)\mathbf{v}(t)\frac{\Delta\alpha}{\Delta t} \tag{2.69}$$

the limit as Δt goes to zero

$$\begin{array}{l} \frac{dq}{dt} = \lim\limits_{\Delta t\to 0}\frac{q(t+\Delta t)-q(t)}{\Delta t} = \lim\limits_{\Delta t\to 0}\frac{1}{2}q(t)\mathbf{v}(t)\frac{\Delta\alpha}{\Delta t} \\ \frac{dq}{dt} = \frac{1}{2}q(t)\mathbf{v}(t)\omega(t) \end{array} \tag{2.70}$$

where $\omega(t)\mathbf{v}(\mathbf{t})$ is the angular rate vector of quaternion Δr which leaves the derivative quaternion state vector in terms of the angular body rates [12]

$$\frac{dq}{dt} = \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.71}$$

From Equation 2.71, it can be seen that when dealing with quaternions, only four integrations have to be done, and the only time trigonometric identities are used are when converting from Euler-angles to quaternions and back. Even though Euler-angles are more intuitive and have one less integration to make, quaternions are: slightly less taxing computationally; do not have a singularity to worry about; and make it easy to model in a virtual world since most computer packages use quater-

nions or angle/axis representations (which are taken directly from quaternions). In order to make the models represented in this thesis more versatile and more robust, quaternions will be used.

2.3 Satellite Three Axis Control

There are three main devices that are used to control the attitude of a spacecraft: thrusters, momentum wheels, and control moment gyros. SimSat, like other actively-controlled satellites, uses two of these three for three axis control: thrusters and momentum wheels. By using three axis control, the operator can actively change the orientation of SimSat by accessing the thrusters and/or each of the three momentum wheels that are attached to the three principal axes. Having three axis control is important so that a satellite can adapt to changing mission requirements. Following is a short discussion on how thrusters and momentum wheels work in order to change the orientation of SimSat.

2.3.1 Thrusters. Thrusters apply torque to a satellite in order to change its orientation by ejecting some mass from a nozzle via pressurized cold gas propellents, hot gas from chemical reactions, or ionized gas from electrical thrusters such as ion or pulsed plasma. SimSat uses cold gas jets vented through nozzles that are attached to SimSat's principal axes, much like in Figure 2.7 below.

Assuming the principal moments of inertia are I_{11},I_{22}, and I_{33} about the b_1,b_2, and b_3 principal axes respectively, then firing the two thrusters attached in the b_2 direction as shown in Figure 2.7 would result in a torque

$$\mathbf{M} = 2\mathbf{r} \times \mathbf{F} \tag{2.72}$$

Where $\mathbf{r}$ is the distance from the center of mass to where the thruster is located on the satellite and $\mathbf{F}$ is the force of the thruster. Most thrusters are fired in impulses

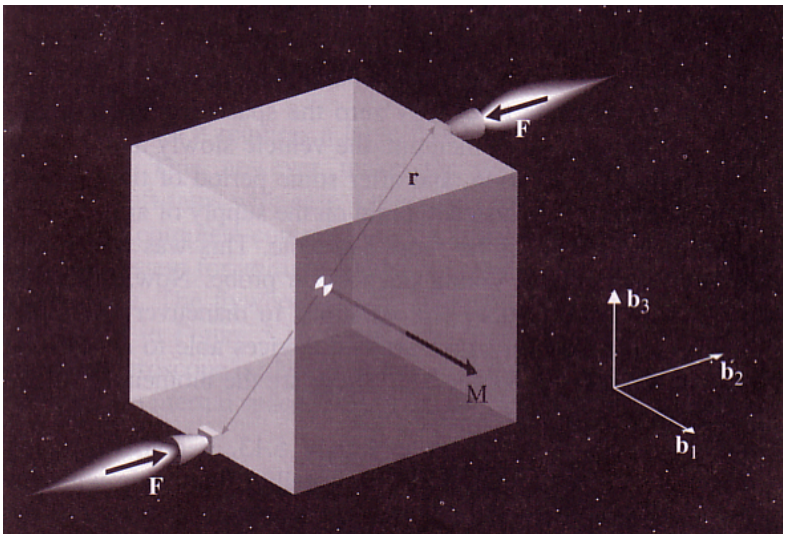

Figure 2.7: Thruster-Controlled Spacecraft [18]

or short durations of time, Δt. Assuming an initial angular momentum of zero, the angular momentum due to an impulse is

$$\mathbf{H} = \mathbf{M}\Delta t = 2\mathbf{r} \times \mathbf{F}\Delta t \tag{2.73}$$

in the b_1 direction. Since $\overrightarrow{\mathbf{H}} = \mathbf{I}\overrightarrow{\boldsymbol{\omega}}$ from Equation 2.12, the satellite gains an angular rate of

$$\omega_1 = \frac{|\mathbf{H}|}{I_{11}} \tag{2.74}$$

about the b_1 principal axis.

A major disadvantage to thrusters is that it is necessary to keep a supply of fuel on-board the satellite. Once the fuel is depleted, control via the thruster is no longer possible. Therefore thrusters are normally used for larger slew maneuvers and momentum wheels are used for finer pointing of the satellite so as not to waste unnecessary fuel. Another disadvantage to thrusters is that as fuel is depleted, the moments of inertia of the satellite change. In most cases, such as Equation 2.74,

the loss of fuel is small and is not normally incorporated into the spacecraft models which assume that the moments of inertia are constant.

2.3.2 Momentum Wheels. Momentum wheels are the main method that SimSat uses to control its orientation. SimSat has three momentum wheels so that their spin axis is mounted parallel to each of the three principal axes. The momentum wheel is nothing more than a flywheel with moment of inertia I_f that is mounted rigidly to the satellite. Figure 2.8 below represents such a flywheel, attached so that it spins around an axis that is parallel to the b_1 principal axis.

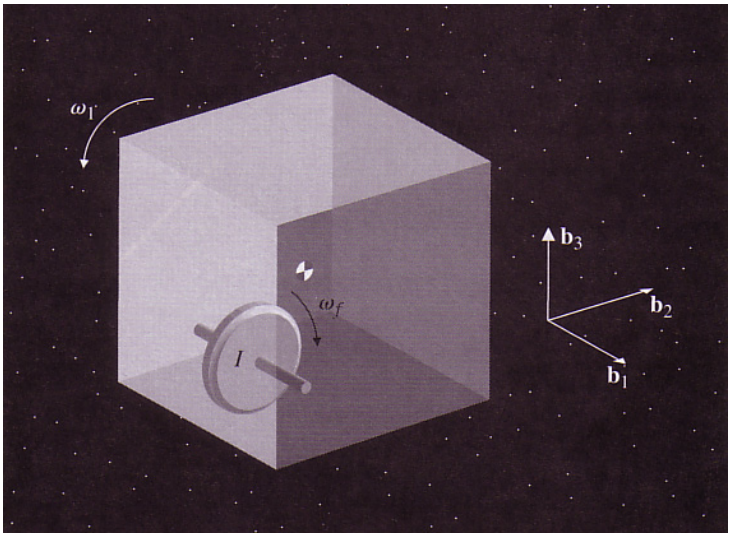

Figure 2.8: Spacecraft with Momentum Wheel [18]

Starting with the flywheel and satellite not moving, the total angular momentum of this system is initially zero. As Figure 2.8 points out, an electric motor spins up the flywheel at an angular speed w_f. In order to conserve angular momentum, the satellite spins in the opposite direction with an angular speed of w_1. Conservation of angular momentum dictates

$$H_{tot} = 0 = I_f\left(\omega_f - \omega_1\right) - I_{11}\omega_1 \tag{2.75}$$

solving for w_1, the spacecraft angular velocity,

$$\omega_1 = \frac{I_f \omega_f}{I_{11} + I_f} \tag{2.76}$$

about the b_1 principal axis.

Typically, $I_{11} >> I_f$, so that the system is sensitive enough to null small spacecraft rates with ease [18]. This sensitivity allows pointing accuracies to about $\pm 0.01^\circ$.

2.4 Summary

SimSat is a rigid body that has three rotational degrees of freedom. The rotational properties of SimSat are nestled in its moments of inertia. Given the moments of inertia and the rates at which the body rotates, it is possible to define the angular momentum of the satellite. Conservation of angular momentum is the basic principal which governs the three-axis controls that are used. Controlling the angular speed of the momentum wheels and the impulses provided by the thrusters enables an operator to control the angular speed of the satellite. It is necessary to transform the angular speed of the satellite about its body axes to an inertial point of view so that the operator can see how the satellite is positioned. This can be done either of three ways, direction of cosines, Euler-angle sequences, or quaternions. Integrating the position from direction cosines is computationally taxing. Euler-angle rotation sequences are intuitive and only have three integrations but include a singularity which must be carefully avoided. Quaternions are the least computationally intensive, have no singularities, and is a form that is friendly for the creation of visually oriented models.

III. Test Set-Up

This chapter reviews the test-setup that is used for this thesis. It covers the hardware used, including the installation of a new fiber-optic gyro. The software used to model and run the tests are then presented. This chapter concludes with an in-depth discussion on the model that provides satellite orientation from received telemetry data.

3.1 Hardware

This first section talks about the hardware that was used in order to run the experiments. There were three main hardware items: SimSat, a Dell computer which acted as the ground station, and a personal computer which was used for processing of the data. In addition to these three main items, there is also the installation of a gyro upgrade that was done on SimSat.

3.1.1 SimSat. SimSat is the main piece of hardware used in support of this thesis. The telemetry data of the control inputs received from SimSat as it maneuvers will be used to generate a visual model depicting attitude determination. A picture of SimSat as set-up in the lab can be found in Figure 1.2.

SimSat was originally designed and constructed by five of AFIT's 1999 Systems Engineering Masters students as their Master's thesis. SimSat was developed in response to the need to simulate satellite behavior with as much fidelity as possible [2]. It now serves as a satellite system simulator and experimental testbed for future Air Force related research topics. SimSat is a very complicated system allowing full-state feedback in terms of control input states and orientation information such as its angular position and rates. In the following pages, those physical characteristics of SimSat which are vital to the objectives of this thesis are brought to attention. A full and detailed specification can be found in the original 1999 thesis [2] and

the theses that followed: one which upgraded SimSat's reaction wheels [3], added a set of cold gas jet thrusters [6], and eventually equipped a thermal ccd camera to SimSat [11].

SimSat's physical dimensions are about 72x21x14 inches in size with an approximate weight of 250lbs. There are three main sections to SimSat: the air-bearing, the three-axis control devices, and the power and communications link. These are pointed out in Figure 3.1.

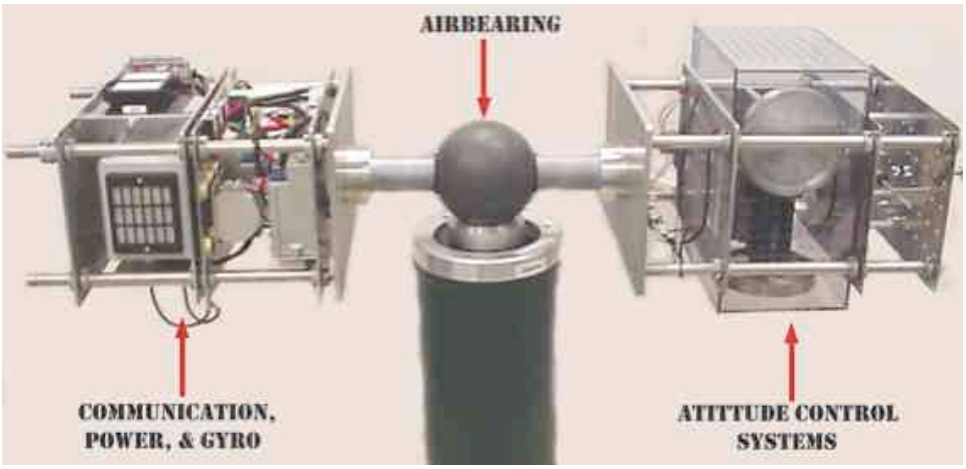

Figure 3.1: SimSat's Main Areas

The first section pointed out is the air bearing. SimSat achieves its three rotational degrees of freedom because its Tri-Axis Spherical air bearing sits upon a cushion of air as shown in Figure 2.1. Six jets, approximately 500 kPa of compressed air, in the air bearing cup produce a less than $12.7\mu m$ air cushion on which SimSat hovers. The pedestal which the airbearing cup is attached limits motion of SimSat in one rotational direction to $\pm 25^\circ$ while providing complete freedom in the other two rotational directions. When SimSat is straight and level, the pitch angle, the angle defined from displacement from the local horizon, is the one limited as is presented in Figure 3.2.

The next area of interest is the section where the attitude controllers are found. There are three momentum wheels that are aligned parallel with each of the assumed

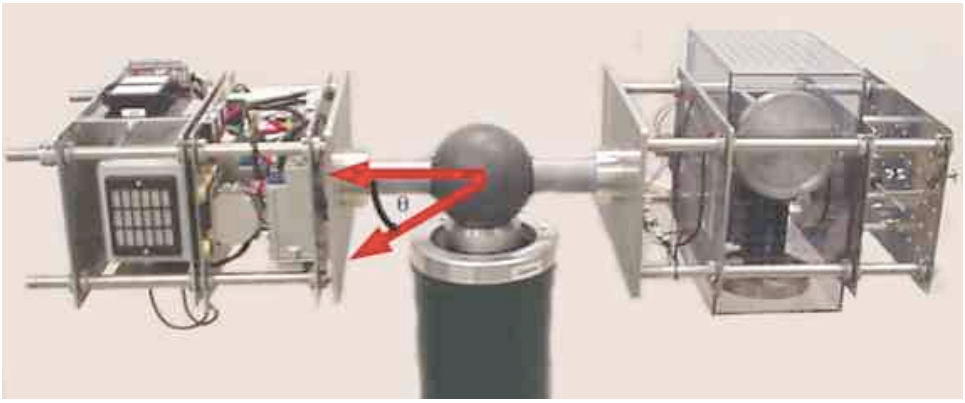

Figure 3.2: SimSat with Pitch Angle and Local Horizon Defined

three principal axes. Unfortunately SimSat is not perfectly balanced. It is assumed for simulation purposes that SimSat is a rigid body and that SimSat's center of gravity is near the center of the spherical air bearing. Instead, the center of gravity is most probably located several centimeters directly below the geometric center because SimSat's structure flexes due to the weight making SimSat a little bottom heavy. This creates an equilibrium position as was noted by Capt French in a 2003 thesis.

> The SimSat actually sags to an equilibrium position, regardless of any attempt to balance it. This obstructs attempts to rotate it to other stable positions about either the pitch or the roll axis, the latter more severely. If one attempts to roll or pitch SimSat to an arbitrary position, it will seek its equilibrium position upon release. This precludes true three-axis control of SimSat [6].

Not only does the off-centered center of mass cause inaccuracies due to this un-modeled force of gravity, the non-symmetric distribution of mass shifts SimSat's principal axes slightly, causing additional inaccuracies that are also unaccounted. These additional inaccuracies are because the input to the 3-axis controllers, i.e. the momentum wheels and thrusters, assume that they are aligned with the principal axes.

There are three momentum wheels mounted in SimSat as found in Figure 3.3. Each of the momentum wheels were fabricated in the AFIT lab. They each have a thin aluminum circular disk with a diameter of $8.625in$ with a steel rim. The moment of inertia of each wheel has been calculated to be $1.955 * 10^{-2} kgm^2$.

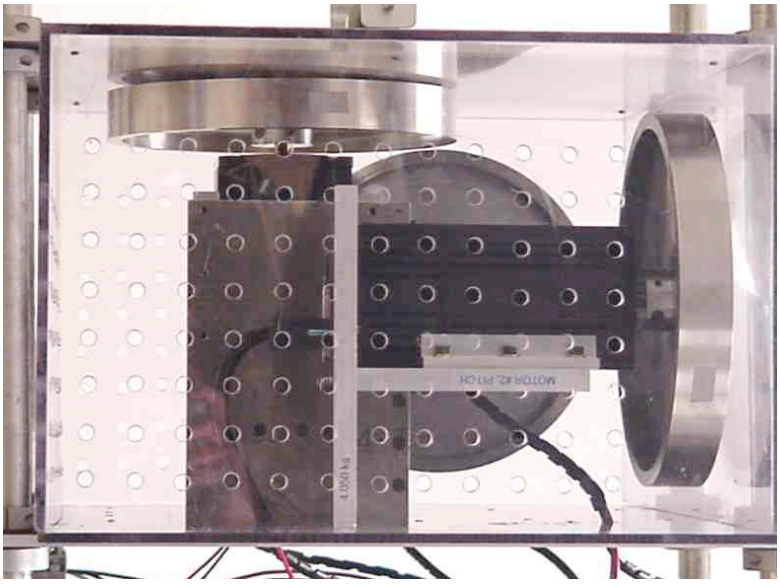

Figure 3.3: SimSat Reaction Wheels for Three-Axis Control

Each of the reaction wheels are spun up by an Animatrics SmartMotor™ Model SM3450 Motor Systems. Each motor system integrates a brushless DC servo motor, motion controller, encoder and amplifier into a single package [3]. Motor characteristics are listed in Table 3.1.

Table 3.1: Animatrics SmartMotor Model SM3450 Motor Systems

Parameter	Value
Weight	2.90 kg
Length	1555 mm
Width	82.6 mm
Voltage	36V
Encoder Resolution	4000 cts/rev
Data Interface	RS232

The other input controller that can be used on SimSat are the thrusters that Capt French installed and tested in 2003 as part of his thesis [6]. The thruster system on SimSat uses nitrogen-based cold gas jet nozzles. Even though SimSat has the ability to use thrusters, the thrusters were not used on SimSat testing because of time constraints imposed by the installation of a new fiber-optic gyro. However, the thrusters were used in the SimSat based simulation that was run.

The last section of SimSat is the power and communications equipment. Three Power-Sonic ® Model PS-12180 rechargeable batteries power SimSat. Each 12 V sealed lead-acid battery has a rated capacity of 18 AmpHours when discharged at the one hour rate. The bus wiring makes 12 V, 24 V, and 36 V available from which SimSat becomes a functional satellite that can receive commands, execute commands, and send telemetry data back. The so-called "brain" of all of this communication and data flow on SimSat is the dSpace ® AutoBox ®.

dSPACE Inc. proprietary hardware and software is used for onboard command, control, and telemetry in real-time. A dSpace ® AutoBox ® DS400 provides the DC computing power and is configured with a DS1005 PPC Processor Board which handles and runs the programs compiled for it by the operator. An operator can upload and download information to SimSat via a RadioLAN® DockLINK™ Model 408-008. This wireless transmitter is utilized for real-time wireless command/data transmission at speeds up to 10 Mbps. A DS2003 32-Channel A/D Board and DS2103 32-Channel D/A Board are used in talking with the Mechanical Gyro and the thrusters. Finally, there is DS4201-S 4-Channel Serial Interface Board that supports RS232 communication at speeds up to 115.2 kBaud. Three serial ports are used for communication to the momentum wheels. The fourth serial port, previously unused, will now be taken up by the installation of the new fiber-optic gyro.

3.1.2 Gyro. The primary gyro used for actual attitude determination in the experiments is a Humphrey model CF-75-0201-1 axis rate gyroscope. It provides

angular velocity and linear acceleration in three body frame axes. However, as noted in Chapter II, only the rotational characteristics of SimSat were used in this experimental set-up since SimSat cannot physically translate. Table 3.2 provides the manufacturer's performance data.

Table 3.2: Humphrey Model CF-75-0201-1 Axis Rate Gyroscope Characteristics

Parameter	Value
Roll Rate Range	$\pm 120°/sec$
Roll Accuracy (Half Range)	$1.2°/sec$
Roll Accuracy (Full Range)	$4.8°/sec$
Pitch/Yaw Rate Range	$\pm 40°/sec$
Pitch/Yaw Accuracy (Half Range)	$0.6°/sec$
Pitch/Yaw Accuracy (Full Range)	$2.4°/sec$

McMaster-Carr Natural Rubber Plate Form Mounts insulate the gyroscope from the main SimSat structure. Figure 3.4 illustrates the installation of the mechanical gyro on SimSat.

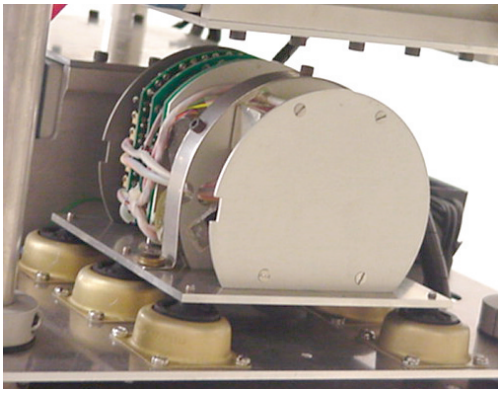

Figure 3.4: Humphrey CF-75-0201-1 Axis Rate Gyroscope and Mounting

Gyro drift has previously been identified as a problem encountered with SimSat. Though the data for rotations around its pitch axis is the most sensitive, all three gyro axes are subject to the drift phenomenon. A study of the drift for the yaw gyro was undertaken by Capt Kimsal in support of his research in 2003 [11]. In his study of gyro rate drift, Capt Kimsal situated SimSat so that it was fixed in

its stand so that no actual movement could take place. Data was then captured at various intervals in order to characterize how the gyro was drifting over time. Figure 3.5 shows the gyro drift rates taken after increasing amounts of warm-up time. 10-minute samples were taken immediately after gyro turn-on ("zero" minutes), after 20 minutes of usage, 50 minutes of usage, and 60 minutes of usage.

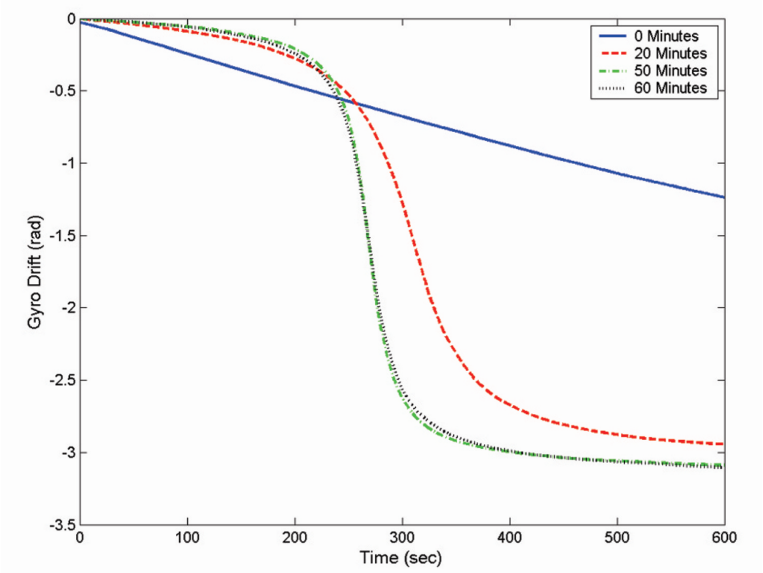

Figure 3.5: Gyro-Drift Data for Yaw Axis [11]

As Capt Kimsal [11] points out,

> As is evident, there is a distinct difference in the behavior of the gyro as it is allowed to warm up. Immediately upon start-up, the yaw gyro exhibits a linear decay in reported angle. As time goes on, it appears to come to a limit; the 50 minute and 60 minute plots lie almost on top of one another.

Capt Kimsal wasn't the only person to note the errors associated with the gyro drifts. Previously, an attempt was made to upgrade this gyro by Capt Dabrowski [3]. A Litton® (now Northrop Grumman® Navigation Systems) model LN-200 Fiber Optic Gyroscope (FOG) was purchased in 2002. Figure 3.6 illustrates the LN-200 FOG.

Figure 3.6: LN-200 Fiber Optic Gyroscope

The LN-200 is space-qualified with up to $1°/hr$ accuracy. This is expected to be a two order of magnitude improvement. It is approximately the same size as the current gyroscope for simple physical integration. Table 3.3 presents the relevant gyroscope characteristics. More data can be found on Northrop Grumman's Website [15].

Table 3.3: Northrop Grumman® LN-200 Characteristics

Parameter	Value
Weight	700 g
Diameter	8.9 cm
Height	8.5 cm
Power Consumption	10 W
Bias Repeatability	1–10/hr
Random Walk	0.04–0.1 $hr^{1/2}$ power spectral density
Data Latency	<1 msec
Data Protocol	RS-485
Data Structure	Synchronous Data Link Control (SDLC)

However, Capt Dabrowski could not get the required information from the gyro necessary to integrate it with SimSat because of data communications problems.

> The fundamental problem is the conversion of the SDLC data stream to an asynchronous data structure for capture by a standard asynchronous RS-485 port. The synchronous SDLC data steam uses a flag/framing structure and therefore has no "start" or "stop" bits allowing the data to remain relatively unaltered within the frame as opposed to an asynchronous structure where the data is usually chopped into 8-bit "chunks". Attempting to receive this synchronous framed data on an asynchronous platform results in data loss where the "start" and "stop" are stripped. The onboard computer is a proprietary design and standard computer cards will not interface with it correctly [3].

The proprietary nature of the gyro made it difficult for researchers at AFIT to take advantage of the Fiber-Optic Gyro until 2003. This is when a board, Figure 3.7, was bought that interfaces the gyro with a RS-232 serial port. Part number: SK-PCB-0201 from SkEyes Unlimited Corporation is a LN-200 interface board that was developed with 3 primary functions:

- Generates the require voltages for the LN-200 IMU from a single 9-18VDC input
- Converts SDLC data packets from the LN-200 into RS-232 serial signal (115.2Kbaud)
- Generates timing pulses synchronized with the LN-200 samples [16].

Figure 3.7: LN-200 Interface Board

The board takes data from the LN-200 and converts it into a serial data stream that can be used by a computer with the correct software implementation in order to read the data from the gyro. The data stream sent from the board is sent out with a frequency of 400 Hz and is made up of packets of data containing 21 bytes with the first byte being a constant header byte. The data format is shown in Table 3.4:

Table 3.4: RS-232 LN-200 Gyro Data Packet

Data Description	# of Bytes
Header (Always $55 hex)	1
X acceleration	2
Y acceleration	2
Z acceleration	2
X rate	2
Y rate	2
Z rate	2
IMU Status	2
Mode/Mux ID	2
Mux data	2
New Flags	1
CRC	1
Total:	21

Knowing this data packet and various scaling equations, which scales the data into units of measure that are useful, a summer intern was able to get the gyroscope working with a pc through a Matlab® software script, gyrorate.m (see Appendix B), on a local pc in the Fall of 2004. Using his work as a starting point, this author was able to build a Simulink® model that could be uploaded to SimSat. Figures and descriptions of models to allow communication to dSpace® from SimSat can be found in Appendix B.

The LN-200 gyroscope was mounted taking the place of the original gyro with the board being attached under the power busses as illustrated by the arrows in Figure 3.8.

As of this writing, the LN-200 gyroscope and interface board were installed but not yet fully integrated with SimSat. All dSpace®/ Simulink® models that currently use the old gyros need to be modified in order to use the new gyros. A rotation matrix needs to be implemented to align the gyro axes with the principal axes. For safety reasons, testing should be done to ensure that the fiber-optic gyroscopes do indeed mimic the output format as the original gyros. Also, further testing needs to be done to try to reproduce random erroneous data spikes that were found in analog

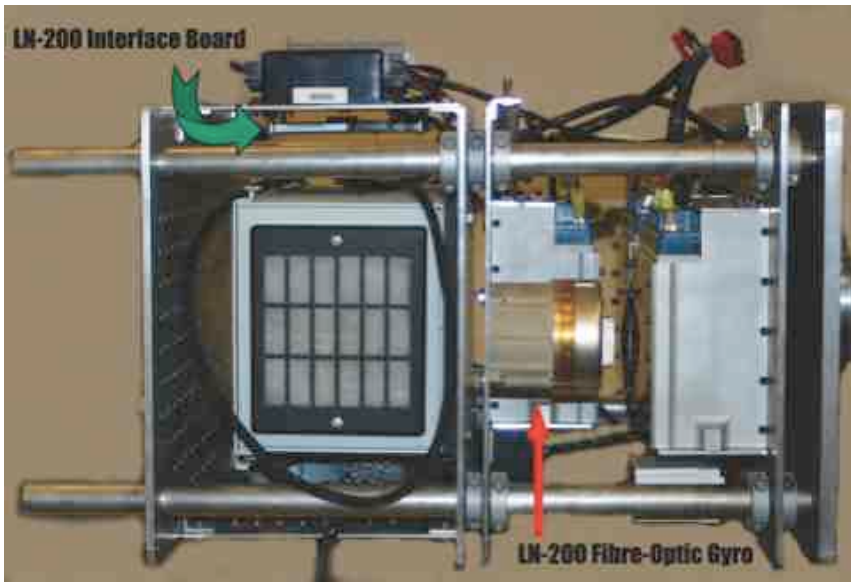

Figure 3.8: LN-200 and Interface Mounting

testing but so far absent in the digital implementation. A low pass filter may need to be implemented to limit the effect of erroneous data.

3.1.3 Ground Station. The ground station of a satellite is the computer that talks to and commands the satellite. For SimSat, the ground station is a Dell 4500 using a 2.26 GHz Intel Pentium 4 with 256 MB of RAM. It runs Microsoft 2000 Professional with Matlab® 6.5/Simulink® 5 along with the proprietary software of dSpace® Controldesk for communication to Simsat from the wireless RadioLAN®. The ground station is used for interactions with SimSat and the capturing of the telemetry data stream. Due to the high learning curve of dSpace® software, this author was unable to get the dSpace® software to work real-time with the Simulink® software on a reliable level for testing purposes. A way was devised to captured the data real-time from SimSat in dSpace® and to export that data to a Matlab® file. From the Matlab® file, the SimSat telemetry data was then converted into signals via a Matlab® script, data.m, to simulate the receiving of telemetry data on an offline computer. In retrospect, this seems to be a preferred way since it makes the attitude determination model more portable. This offline computer happened to be

the author's personal mac, a 1.33 GHz PowerPC G4 12" Powerbook with 1.25 GB DDR SDRAM that was running Mac OS 10.3.7 and the unix environment X11 for Mac OS X. This choice of computer was done for reliability and ease of presentation purposes. However, any pc capable of running Matlab® 7 and Simulink® 6 are able to run the model with graphs as the output. For presentations purposes, the Matlab® Virtual Reality Modeling Language (VRML) toolbox is needed to run the visual model.

3.2 Software

As previously mentioned, there is software required to interact with SimSat and to run simulations. There are three main programs used: dSpace® Controldesk software; Matlab®; and Simulink® with the Real-Time Workshop and VRML toolboxes installed.

dSpace® software is the software that connects the user on the ground station to the Autobox® on SimSat. dSpace® allows software and commands to be uploaded to SimSat and gathers and displays data received from SimSat. The software also acts as an intermediary when dealing with other applications by compiling and building code so that the Autobox® can run the applications built for it. An example of a dSpace® Controldesk layout that is used for monitoring telemetry data is show in Figure 3.9.

The intermediary programs that dSpace® Controldesk work hand-in-hand with are Matlab® and Simulink®. Matlab® is a scientific language package by a company called The Mathworks that is used extensively in engineering and scientific fields. It is a powerful mathematics package based upon the Maple Mathematical Engine. One of the greatest benefits of Matlab® is Simulink®.

Simulink® is a model-based programming tool. It is used to build computer programs using modeling blocks, making sort of a visual based programming language. The benefit of Simulink® is that there are add-on toolboxes that are available

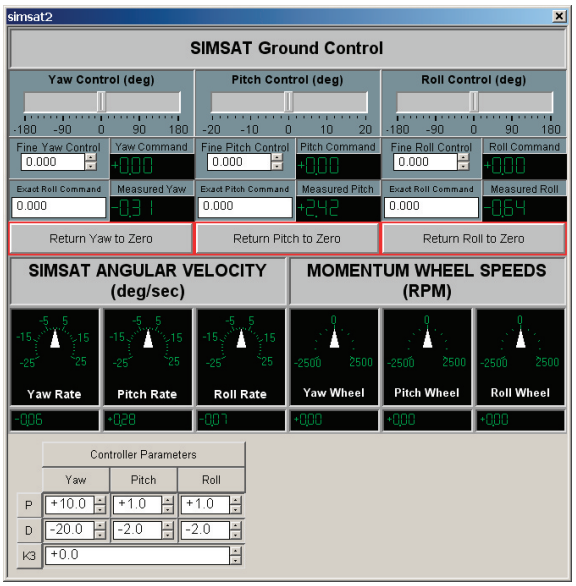

Figure 3.9: Control Desk Layout

for purchase or that can be created by the user. Toolboxes are a set of ready to use function blocks that output values after carryout specific calculations to given inputs.

This thesis requires the use of at least one additional toolbox that is associated with SimSat. That toolbox is the Real-Time Workshop (RTW) Toolbox. The Real-Time Workshop toolbox contains the blocks and the code that allows Matlab® and Simulink® to talk to dSpace® Controldesk, and thus SimSat. It is this toolbox that allows programs and models built in Matlab® and Simulink® to be run on SimSat. This software is found on the ground station computer and is licensed by AFIT.

It should be noted that SimSat uses version 6.5 R13.1 of Matlab® and Version 5 of Simulink® which is installed on the ground station computer. The current versions of MatLab® and Simulink® are 7 R14 and 6 respectively. There are major differences in the different versions, with the new version not being able to down-convert convenient new Simulink blocks to the older version. These enhancements in Version 6 of Simulink® simplifies the process of coding in Simulink® that would involve quite complicated workarounds and previous knowledge in the C or Ada programming languages in downgrading models to the previous version.

This author ran into this incompatible version problem twice during the course of his research. The first time was in the generation of the attitude determination model and the second being the effort required to get the fiber-optic gyro talking with SimSat. Initial work was done in the current version of Matlab® and Simulink®. During the efforts of transporting and downgrading the attitude determination model, the author ran into trouble getting the model to work real-time with the telemetry data in dSpace®. This author believes that the problem did not lie with the downgraded version of the model, but rather the extensive learning curve involved in getting dSpace® to successfully hand off data to the Simulink® model in real-time. After many weeks of struggling to get the models to work nicely together, it was decided, for the sake of progress, to capture the telemetry data to a Matlab® file. The telemetry data was then converted into a time-based signal in Simulink® to simulate the receiving of the telemetry data in real-time.

With the decision made to work off-line, it was then decided to keep and use the Attitude Determination Model that was created in Version 6 of Simulink® since it could be read easier by a new user. This also allowed the model to be run on any computer running the version 7 of Matlab® with Simulink® 6 installed. This author used Matlab Version 7 running in X11 for Mac Os X for offline work.

This decision lead to another, very minor, problem. The visualization software used in the past, RealMotionPC3D, which shows a 3-D model of SimSat orientation over time is proprietary. It worked with dSpace® and not Simulink and was not widely available on other computers. To replace the visualization program, the author chose to use his personal copy of the Virtual Reality (VR) toolbox that integrates with Matlab® 7 and Simulink® 6. A VRML model of SimSat that was used is illustrated in Figure 3.10. VRML is a standard modeling language for virtually reality that was originally created for use with the World Wide Web. The greatest users of this modeling environment include NASA and the United States Navy. Since the toolbox works with Unix, Microsoft, and Mac OS X versions of Matlab® and

Simulink®, there should be minimal effort required in porting the complete Attitude Determination Model with the VRML visualization.

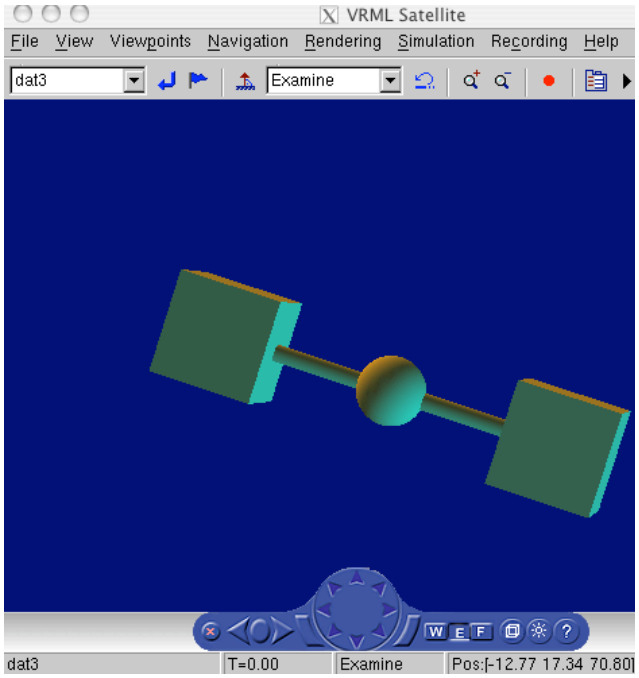

Figure 3.10: VRML Model of SimSat

3.3 Simulation Models

There are three main Simulink® simulation models used for the test-setup in this thesis. The first two models were taken from previous theses unaltered. It will be the results from these simulations which will decide the success of Attitude Determination Model.

3.3.1 PD Dual Controller Simulation Model. The first test done was a Simulink® simulation using a Simulink® Model that simulates applying a PD Dual Controller to a simulated SimSat as created by Capt French in 2003 [6]. This model, top level shown in Figure 3.11, was chosen to be tested first because it uses dual control, reaction wheels and cold gas jet thrusters, and because it is a software simulation.

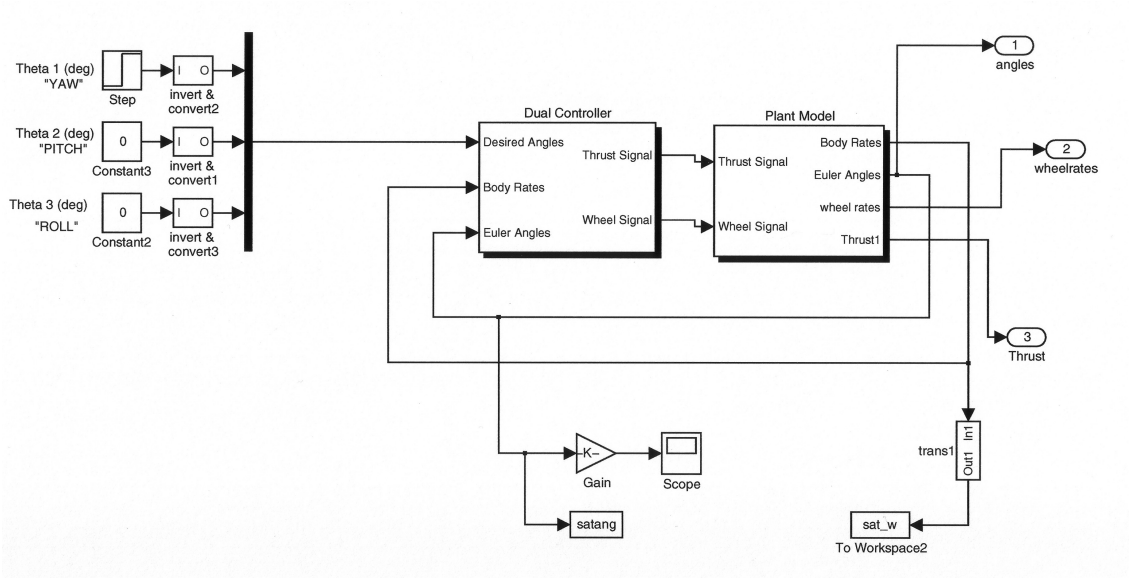

Figure 3.11: Top Level PD Dual Controller Simulink Model

The model as seen in Figure 3.11 is used unmodified as developed by Capt French for his thesis. More information on this model used can be found in [6]. The momentum wheel rates in rpm, the thruster indicator signal, and orientation information from the gyroscope are fed to output ports so that they can be used by the Attitude Determination Model. Tests run from this simulation would serve as validation before attempting to try and integrate the Attitude Determination Model with SimSat.

3.3.2 SimSat MOI Estimation Model. After a successful simulation on the PD Dual Controller Simulation, the next step was to run SimSat and feed the telemetry data to the Attitude Determination Model for real world comparisons. For this next part of testing, a SimSat model created by Capt Dabrowski in 2003 to estimate MOI for detection purposes was selected [3]. This model, top level shown in Figure 3.12, was chosen because it was a simple model which had a single commanded step input. More information on this model used can be found in [3].

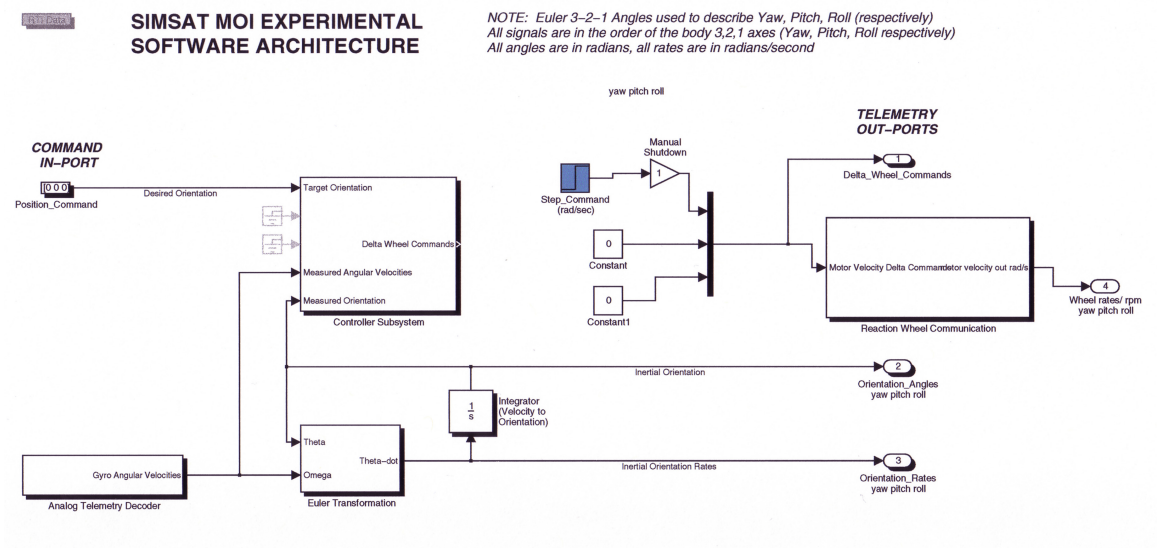

Figure 3.12: SimSat MOI Test-Setup

The model as seen in Figure 3.12 is used with only slight modifications to the step input. The step input was changed to test each of the principal axes individually, and then simultaneously. Gains were applied so that SimSat would avoid hitting the pedestal due to clearance limitations. Table 3.5 shows the different gains applied to each direction used in testing.

Table 3.5: MOI Test Matrix

Data Set Name	Yaw Gain	Roll Gain	Pitch Gain
Yaw alone	1	0	0
Pitch alone	0	0	.2
Roll alone	0	1	0
All	0.8	0.2	0.2

Leads from the momentum wheel rates and orientation information from the gyroscope are fed to output ports so that dSpace® will perceive that information as telemetry data from SimSat. The telemetry data would be captured by dSpace® and put into a Matlab® formatted file so that they can later be accessed by the Attitude Determination Model.

3.3.3 Attitude Determination. The Attitude Determination Model starts with a top level block in Simulink® as shown in Figure 3.13. Its two main inputs are the momentum wheel rates and the thruster on/off indicator signal. Also fed into this model are the Euler orientation angles that are used for setting the initial orientation condition and for model comparison.

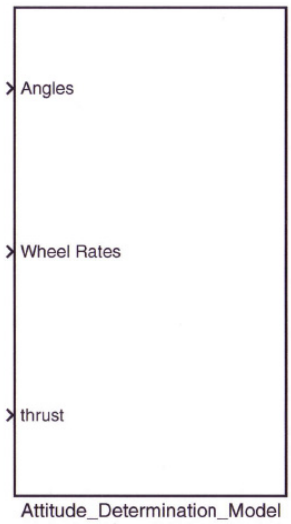

Figure 3.13: Attitude Determination Model -Top Level-

The next level of the attitude determination model, illustrated in Figure 3.14, contains the connections to the visual outputs of this model. Various scopes are used to compare the time histories of the quaternions and the Euler-angles for both the model and actual data from the gyro.

The first block of interest is labeled "Initial Reference." This block takes the incoming Euler-angle sequence from the gyros and outputs a vector of quaternions using Equation 2.61. This vector of quaternions gets fed to the Attitude Determination Block for use as the initial quaternion state. It also gets passed on to the Virtual Reality block for a visual representation of the Satellite maneuvering. The original signal of Euler-angles also get sent through for comparison with the Euler-angles generated from the Attitude Determination block.

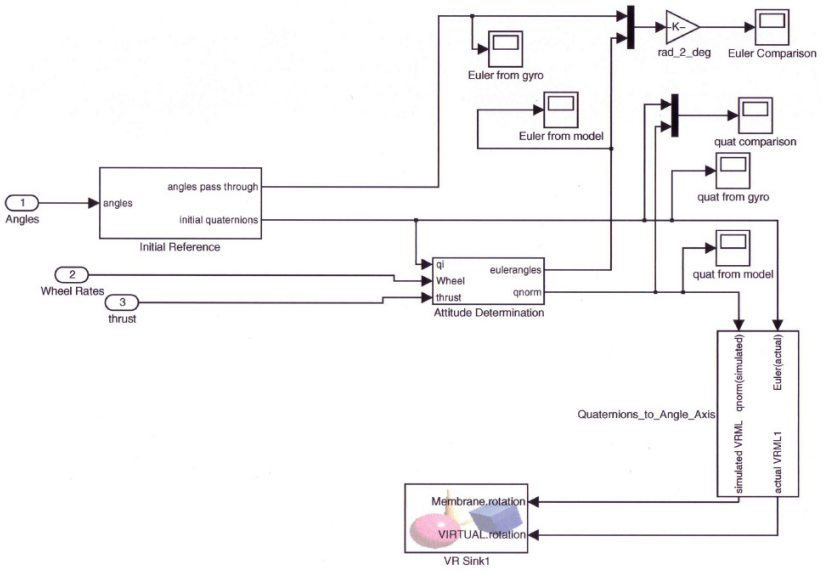

Figure 3.14: Visual Level of Attitude Model

The next important block in Figure 3.14 is the Quaternions_to_Angle_Axis block. This block takes the quaternion representation of SimSat from both the gyros and from the Attitude Determination block and retrieves the angle, the angle associated with q_0, and the axis about which that angle is rotated, the unit vector (q_1, q_2, q_3), directly from the quaternion vector $\mathbf{q}$. From here, the angle/axis representations of the gyro-based SimSat and the controller-determined SimSat are received by the VRML model and can be viewed real-time as the simulation is taking place.

The final block in Figure 3.14 is the heart of the Attitude Determination Model, the Attitude_Determination_Block. This block is broken up into two main parts: transforming the controller telemetry data into body axis rates, Figure 3.15; and transforming the body axis rates into quaternions and integrating to get the modeled orientation of SimSat in both quaternions and Euler-angles, Figure 3.17.

Figure 3.15 shows the information that must be known in order to determine the attitude of a satellite given the momentum wheel rates and the thruster on/off indicator signal. The inertia of the reaction wheels and the principal moments of

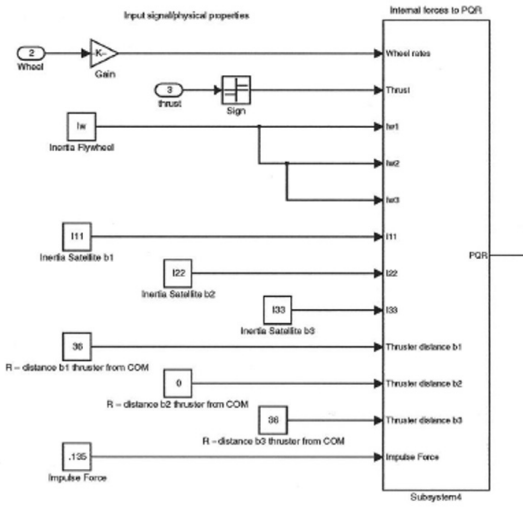

Figure 3.15: Attitude Determination from Control Inputs

inertia of the satellite need to be known. In addition, some additional information needs to be included for the thrusters, such as the force of the impulse and the distance of the thruster from the center of mass along the principal axis. This information is assumed known either by calculation or direct measurement and Table 3.6 lists the values used for this experiment.

Table 3.6: Initial Conditions

Parameter	Value
I_{wheel}	$2.08 lbs * ft^2$
I_{11}	$91.42 lbs * ft^2$
I_{22}	$957.61 lbs * ft^2$
I_{33}	$960.68 lbs * ft^2$
Impulse Force	0.135 lbs thrust
Distance from $\mathbf{b_1}$	36 in
Distance from $\mathbf{b_2}$	36 in
Distance from $\mathbf{b_3}$	36 in

The principal moments of inertia for SimSat were determined experimentally through conservation of angular momentum:

$$I_{ii} = \frac{I_{rw|i}\Delta\omega_i}{\Delta\Omega_i} \tag{3.1}$$

where i is the body axis number. Appendix A outlines the test procedure used.

The data in Table 3.6, along with the reaction wheel data, in rpm, and the thruster indication signal, all go in the Internal Forces to PQR block which is shown in Figure 3.16. In the top block of Figure 3.16, Equation 2.76 is used in determining the angular rates of the satellite around the three body axes from the principal moments of inertia and the reaction wheel rates.

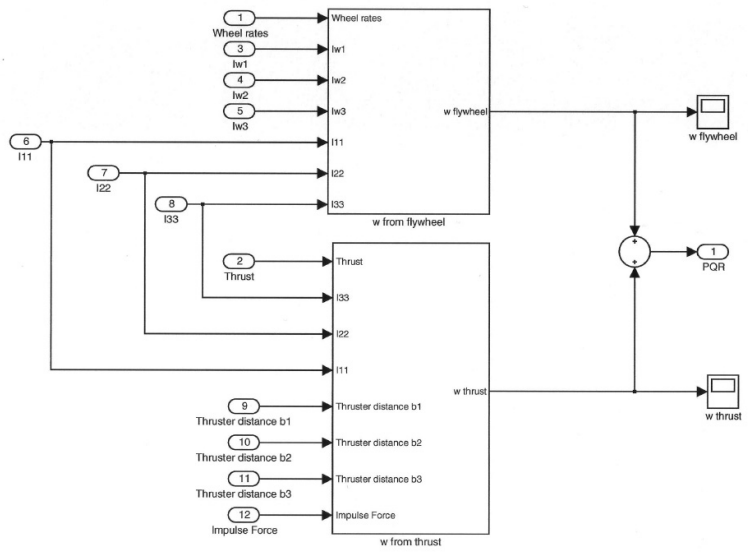

Figure 3.16: From Wheel Rates and Thruster Data to Body Orientation Rates

The bottom block of Figure 3.16 uses Equation 2.74 in determining the the angular rates of the satellite around the three body axes due to the moments of inertia and the data from the thrusters. Then the angular velocities due to the reaction wheels and the thrusters are added together resulting in the total angular velocity of the satellite in terms of the three body axes. This total angular velocity is

then passed to a block which calculates the derivative of the quaternion, $\dot{q}$, as shown in Figure 3.17.

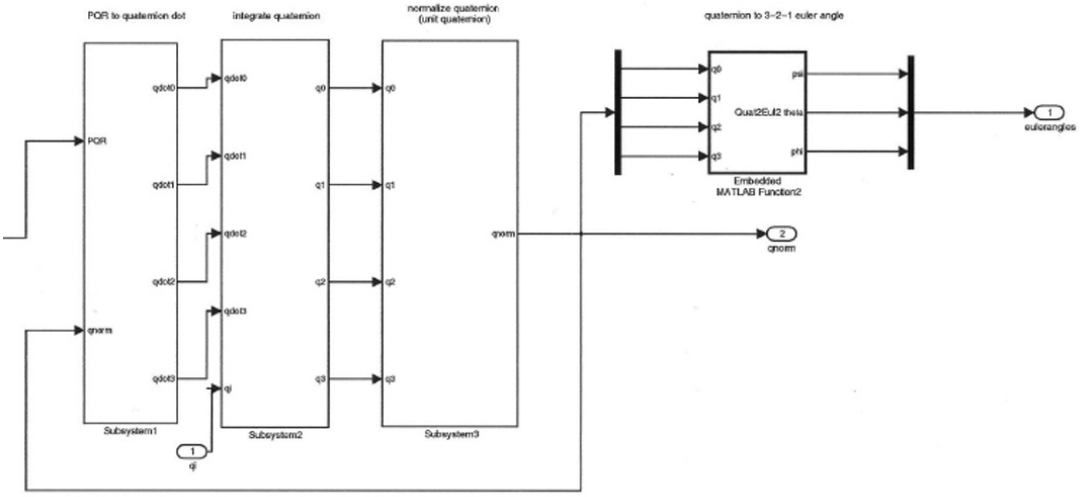

Figure 3.17: From Body Orientation Rates to Quaternions and Euler-Angles

The quaternion rate, $\dot{q}$, is calculated by Equation 2.71 given the satellite body axis angular velocities and the current position of the satellite in terms of quaternions. Next, $\dot{q}$ is integrated using a discrete integrator with the initial condition being the first quaternion position of the gyro so that both the gyro and the model start from the same orientation. Assuming an Autobox®-limited time integration step of 0.05 seconds, the zero-order hold discrete time integrator used is found in Equation 3.2:

$$\begin{array}{rrcl} & \mathrm{y(n)} & = & \mathrm{y(n-1)+T*u(n-1)} \\ \textit{Let} & \mathrm{x(n+1)} & = & \mathrm{x(n)+T*u(n)} \\ \text{Step 0 :} & \mathrm{y(0)} & = & \mathrm{x(0)=IC} \\ & \mathrm{x(1)} & = & \mathrm{y(0)+T*u(0)} \\ \text{Step 1 :} & \mathrm{y(1)} & = & \mathrm{x(1)} \\ & \mathrm{x(2)} & = & \mathrm{x(1)+T*u(1)} \\ \text{Step n :} & \mathrm{y(n)} & = & \mathrm{x(n)} \\ & \mathrm{x(n+1)} & = & \mathrm{x(n)+T*u(n)} \end{array} \tag{3.2}$$

The third step in the series is normalizing the quaternion, this is achieved with Equation 2.53. A normalized quaternion, which indicates the satellites current position, then gets sent into three directions. A normalized quaternion is looped back into the first block of Figure 3.17 for the calculation of $\dot{q}$. A normalized quaternion is also output for visualization purposes. Finally a third normalized quaternion is sent through a function that uses Equations 2.62, 2.63, and 2.64 to transform the quaternion back to a 3-2-1 Euler-angle sequence.

3.4 Summary

This chapter reviewed the test set-up for determining if attitude controllers can be used for attitude determination. The main features of SimSat were described, as were the other hardware and software used for testing. Characteristics of the mechanical gyro that was used for testing and the integration of the new fiber optic gyro was discussed. Captains French's and Dabrowski's models, which are used in the verification of the attitude controller-based determination model, were reviewed. This chapter concluded with the main portions of the attitude determination model that was created to determine a satellite's attitude from its telemetry data. As currently developed, the attitude determination model does not include external disturbances and loss mechanisms such as gravity gradient torque and frictional losses from air and the bearing. The effects of these will be explored in the testing discussed in the following chapter.

IV. Simulation and Experimental Results

This chapter reviews the data from the SimSat simulations and experiments of the attitude controller-based determination model. The impact to testing from the current mechanical gyros is discussed. The drift improvements of the new fiber-optic gyros are reviewed. And finally, the feasibility of a torque-free attitude controller-based determination model being implemented is assessed.

4.1 Model Simulations

Before SimSat testing commenced, a software simulation was used to verify the attitude determination model which would be used in conjunction with the telemetry data of SimSat. The software simulation was taken from Capt French unmodified. This simulation was chosen since it modeled SimSat, including its thruster and reaction wheel control systems, in a torque-free environment.

The reaction wheel speed, in rpm, and the thruster on/off indicator signals were inputted into the attitude determination model, simulating a telemetry downlink. A step input was applied and the simulation started. Capt French's simulation of SimSat inserted a step input resulting in a 60° maneuver. Initiated by thrusters commanding a large slewing maneuver, the simulated SimSat rapidly yawed to about 60° before the reaction wheels kicked in for finer accuracy to settle SimSat at the desired 60°. Figure 4.1 compares the attitude from Capt French's model to that of the attitude controller based model.

As illustrated in Figure 4.1, it is hard to distinguish between the simulated SimSat and the model derived from the controllers, especially during the controlled maneuvers. These results verified that the model created to determine attitude from telemetry data acquired from attitude controllers, such as thrusters and reaction wheels, is correctly defined for a torque-free environment.

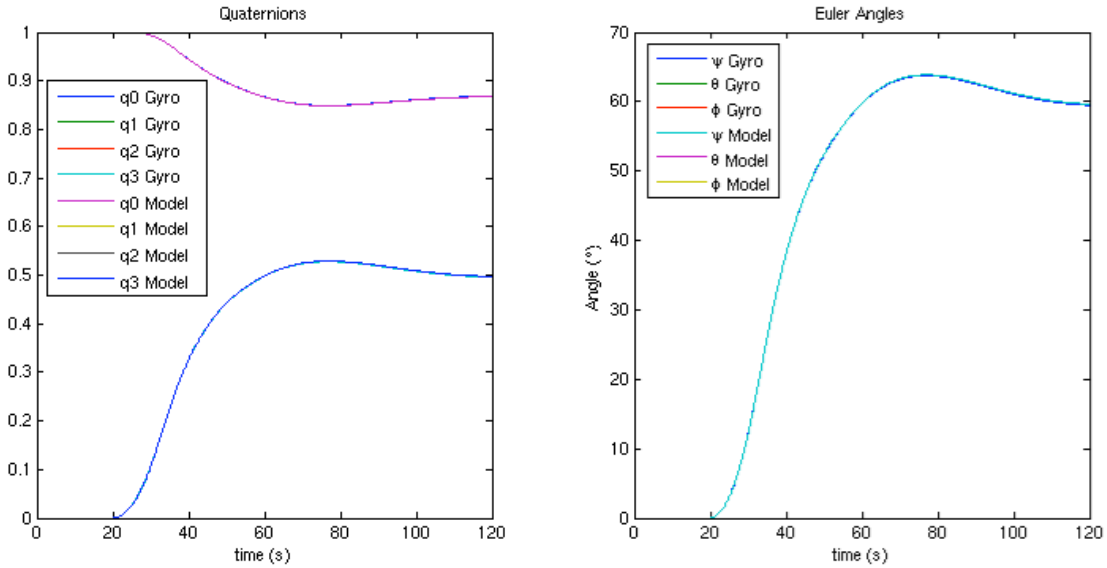

Figure 4.1: Simulated Attitude Determination from Thruster/Reaction Wheel Acquired Telemetry Data

One important premise that is clearly evident from taking a closer look at the last ten seconds of Figure 4.1 is that the system must be well-known in order for the attitude controller method to work. Figure 4.2 highlights a separation between the simulation and the attitude model.

Even though Capt French's model does not include external forces, it does include an estimated internal energy model. Because of this energy model, there is an an energy loss due to the attitude controllers which leaves a small amount of momentum build-up in the reaction wheels. This momentum build-up leaves the simulated SimSat with a small bias in the reaction wheel speed when SimSat comes to a rest. This bias was not present at the start of the simulation. Due to this bias, the attitude controller-based determination thinks that SimSat is still spinning. This energy loss is not built into the attitude controller-based determination model.

In order for an attitude controller-based determination model to be reliable, the satellite's systems and surrounding environment needs to be well known and included in the model. External forces may not have been included in the model,

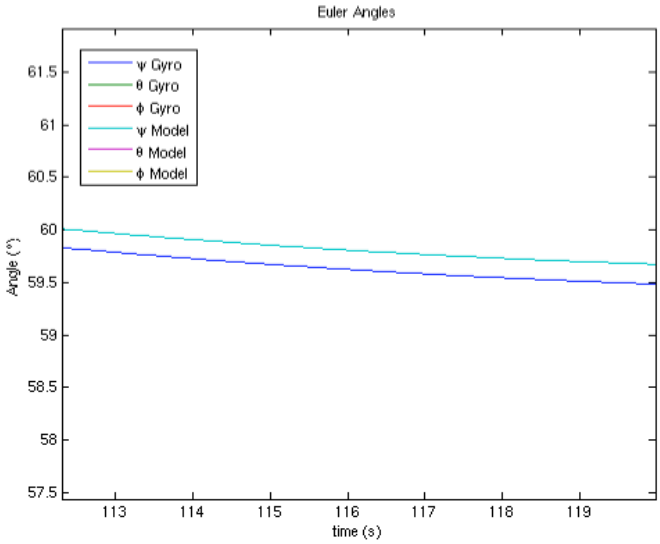

Figure 4.2: Separation Due to Un-Modeled Energy Loss

but they will have the same effect as the internal energy loss shown in Figure 4.2. They will need to be known so that their effects can be accounted for.

4.2 Gyroscope

As seen in Chapter 3, Figure 3.5 illustrates the drift rates of SimSat's mechanical gyros. This drift will lead to inaccuracies when trying to compare the orientation from the gyros to the attitude controller-based orientation. In addition to the errors from assuming a torque-free environment, the gyro drift decreases the accuracy of the assumed SimSat position and therefore limits the effective testing time to below 30 seconds.

In Figure 3.5 Capt Kimsal noted trends in the yaw gyro data (due to thermal heating of the mechanical gyro) so that he might filter out the effects of the drifting from the data. There appear to be trends in the data that last on the order of 10 minutes. However, as he noted, it did not seem to work as well as he thought [11]. Since testing time was below 30 seconds and a new fiber-optic gyro was working with SimSat, drift data for all three axes for both the mechanical and the fiber-optic gyros

was captured at 2 minute intervals, 10 minutes apart, in order to determine if the thermal effects are repeatable on a shorter time scale for the mechanical gyro and if they exist on the new fiber-optic gyro. The amount of drift, measured in degrees, is plotted against 120 seconds of time in Figures 4.3, 4.4, and 4.5 for the yaw, pitch, and roll directions respectively.

The maximum drift for the fiber-optic gyros is about 1° in 2 minutes as shown in Figures 4.3, 4.4, and 4.5. From a cursory look from the three graphs, it is easily seen that this is a great improvement over the existing mechanical gyros. In addition to comparing the new fiber-optic gyros to the old mechanical gyros, the purpose of retesting the drift rates was to see if any trends existed on a smaller time table for use in analysis of the results of the attitude determination model.

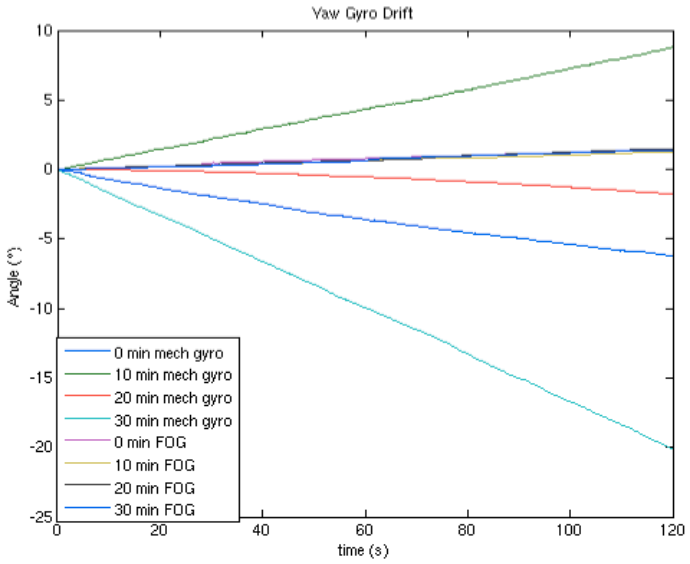

Figure 4.3: Yaw Gyro Drift Rates

Looking at the gyro drift from the yaw gyro in Figure 4.3, there does not appear to be any trend associated with the data. The data does appear to show that the drifting is linear, but this contradicts the nonlinear portions of data from Figure 3.5. No time based trends are noticed with the data capture at the zero time mark, which shows a negative data drift, while the ten minute sample shows a

positive drift rate. The values of drift vary from about 2° to 20°. However, the small number of tests ran does not preclude the yaw gyro from reaching an even higher drift rate, such as the 25° drift in roll found in Figure 4.4.

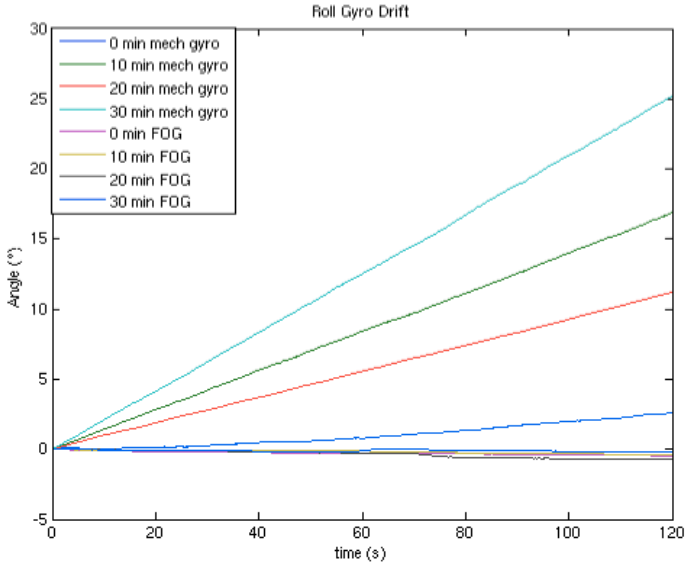

Figure 4.4: Roll Gyro Drift Rates

Figure 4.4 features the same characteristics as Figure 4.3. It shows that the mechanical gyro in the roll direction can potentially reach drift rates of over 10° per minute. Rates as high as this predict that the data from the gyro will very quickly get out of sync with what is actually occurring. For testing in one rotational direction, this would allow reasonable results for the attitude determination model at approximately 20-30 second testing intervals. However, for rotations tracking the three rotational degrees of freedom, this means that model verification could prove to be difficult in tests longer than 10 seconds due to the inaccuracies of the gyros' representation of SimSat's true orientation.

Figure 4.5 represents the drift from the pitch gyro. It too shows no deterministic trends, but does show that the gyro drift is not necessarily linear in small time samples. The non-linear drifts are seen at the 10 and 30 minute time samples in Figure 4.5. In addition to this drift error, which cannot be seen from the graphs, is

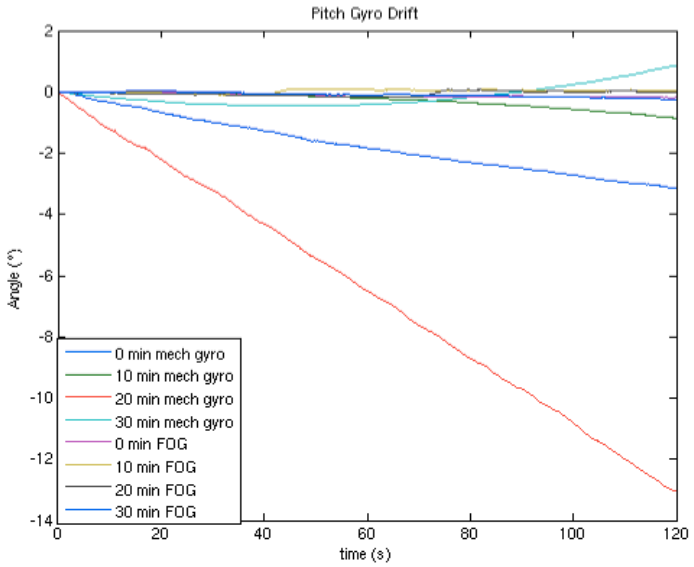

Figure 4.5: Pitch Gyro Drift Rates

that the accuracy of the gyro in this direction is worse than the other gyros by at least a factor of two. A gain of 10 is also included in the pitch gyro compared to the other two rotational directions, which multiplies any errors that do exist. The mechanical gyro was intentionally installed in the orientation for which the worst gyro detects the pitch rate because of the limited movement in the pitch direction as pointed out in Figure 3.2. From these inaccuracies in the pitch gyro, it is assumed that data will be the least accurate in the pitch direction.

4.3 SimSat Results

This section discusses the results of the SimSat testing of the attitude controller-based determination model using the reaction wheel telemetry data acquired from a SimSat downlink. Four main experiments were run with each experiment run about five times over a period of a week. The four main experiments included the three principal axis specific maneuvers and one experiment that tested all three rotational degrees of freedom. All experiments ran the same simulation model.

The model used was created for estimating the principal moments of inertia of SimSat as designed by Capt Dabrowski. This test was chosen because it has a quick, predictable, and repeatable reaction wheel rates which provide enough torque to get the best possible data with an attitude determination model that assumes no external forces exist in an inherently force-based environment. Un-modeled forces, such as gravity, drag, possible air currents [11], and the errors due to gyro drift will affect the results of the attitude determination model when it is compared with the mechanical gyro-based orientation of SimSat.

4.3.1 Yaw. The first tests began with a yaw maneuver. This rotation is thought to be the easiest to model. With SimSat being a little bottom heavy (from the flexibility of SimSat's structure), causing an external torque due to gravity, and the pitch gyro errors high, the yaw direction looks as if it has the best chances for success. The yaw maneuver rotates SimSat in a plane that is perpendicular to gravity, thus eliminating the main external force known for testing. Figure 4.6 is an example of the typical results from the SimSat testing. Position is shown in terms of quaternions and Euler-angles.

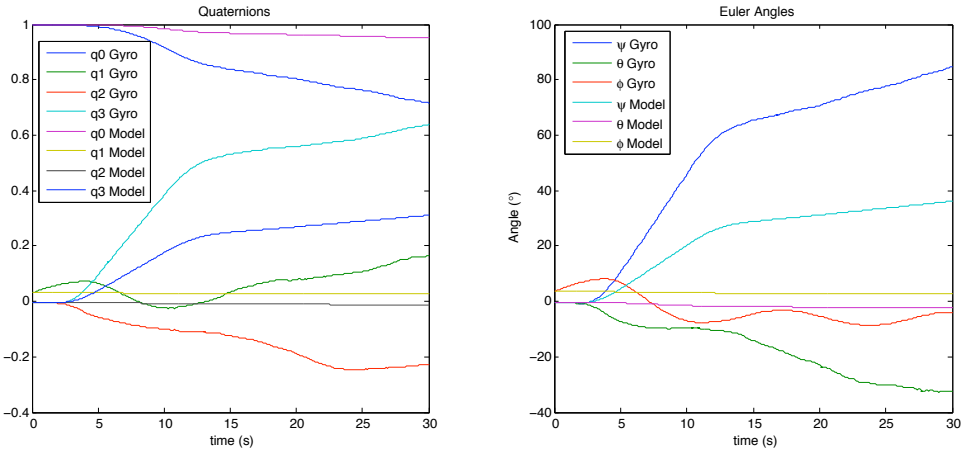

Figure 4.6: Orientation Determination in the Yaw Direction

The attitude controller-based determination model uses quaternions to describe the rotations. However, when dealing with just one rotational degree-of-freedom

around a principal body axis, it is visually pleasing to see the data in Euler-angles since each rotation deals with only one angle and thus only one line on the plot. As the left graph of Figure 4.6 illustrates, quaternions will have at least 2 lines of interest for the same data; q_0, the angle of rotation; and either q_1, q_2, or q_3, the axis (unit vector) it is turning about. Since quaternions and Euler-angles show the same information, Euler-angles will be used throughout the rest of this discussion for less taxing visualization purposes.

The first thing to keep in mind when looking at these graphs is that the reference point for the attitude controller-based model is from the initial point of the mechanical gyro. This is a major drawback of attitude controller-based determination. There is no way to actively retrieve a reference starting point. Information from other systems will need to be used unless passive external torques, such as gravity or the earth's magnetic fields, are built into the model in conjunction with attitude controllers sensitive to those passive external torques. Otherwise, controller-based attitude determination must integrate to get position from accelerations and velocities associated with the controller's method of producing torque.

Looking at the Euler-angle plot of Figure 4.6, the pitch and roll mechanical gyro errors immediately stand out. ϕ, the roll angle, looks like it is drifting slightly and as though it has a small oscillation to it. This oscillation is not due to the mechanical gyro drift, but more likely a combination of SimSat's MOIs not being perfectly symmetric and aligned with the principal axes, i.e. a stable point due to the center of gravity being below the geometric center. It is a small oscillation and has little impact to testing. An oscillation can also be seen in the pitch angle, θ, but is overpowered by the drifting and accuracy errors in the gyro for pitch. θ has deviated almost 40° in the thirty seconds of testing, but since pitch is limited to about 30° due to test set-up, it is physically impossible for SimSat to be in the position as shown in Figure 4.6. As is representative of Figure 4.7, the extremely bad errors in pitch show up in all of the tests that were done.

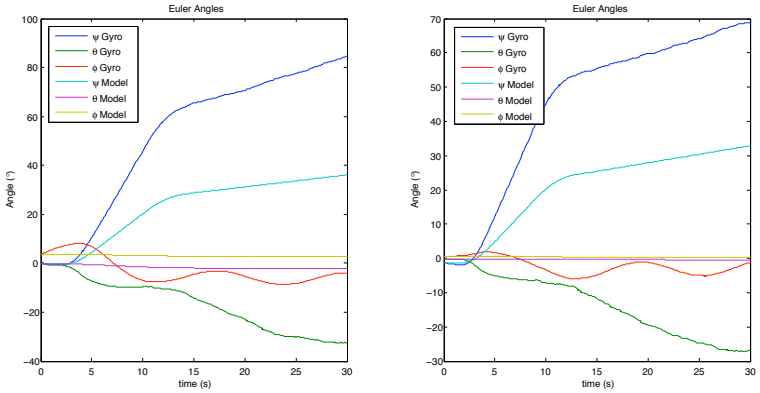

Figure 4.7: Two Yaw Test Runs

Figure 4.7 shows two tests with the same exact setup. Similarities in the pitch and roll data can easily be seen. Also, there are similarities seen in the yaw data itself. Figure 4.7 shows that the yaw tests were repeatable. On first glance it looks as though the data does not match, but looking closer, similar slopes and shapes of the yaw gyro and model data seem to indicate that there is a gain error of about 2 in the model. However, after double checking the MOI calculations and units, there is no apparent error in the model. Figure 4.8 shows a yaw test, with the pitch and roll data zeroed out. The left is depicted as the model dictates, and the right shows the same test with the moment of inertia of the reaction wheel increased to 4.75 lbft^2, a little over double the actual calculated MOI of the wheel.

The figure on the right of Figure 4.8, shows that the increase in the moment of inertia of the wheel aligns the yaw data almost on top of one another and tracks fairly well for all of the thirty seconds with most of the separation occurring at the end of the test due to the mechanical gyro drifting. This news is both good and bad. The good part is that this result is repeatable and thus validates the concept of the model. The bad news is that the source of the error is still unknown. The problem is not the MOI of the wheel, but rather a combination of incorrect principal MOIs for SimSat and the possibility of an un-modeled force. The wheel has been manufactured at AFIT and its MOI properties are known. The MOIs of SimSat are however based

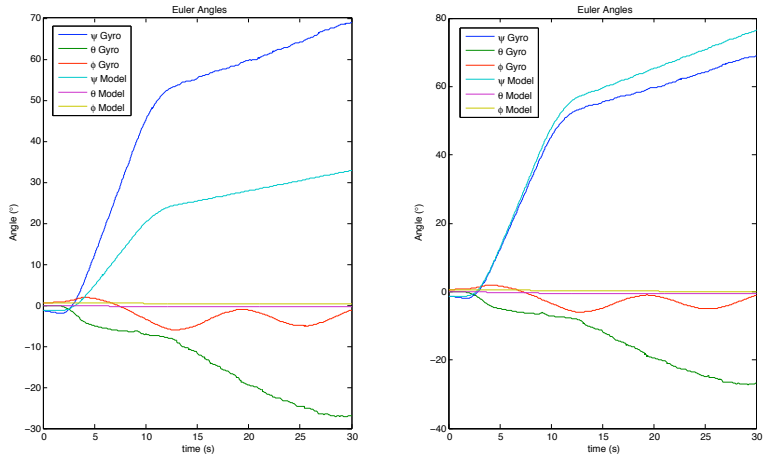

Figure 4.8: Yaw Test with Original(left) and Increased Reaction Wheel MOI(right)

upon an experimental test with bad gyros which increases the likelihood of errors in SimSat's MOI calculations. The ratio of the reaction wheel and SimSat's MOIs is the only mathematical factor in the attitude determination model as seen from Equation 2.76. The other possible source of error is an measurable un-modeled force which previous master's students think is due to an air current in room [11].

4.3.2 Pitch. The yaw data showed promising results and shared some insight on the errors associated with the pitch gyro. From the magnitude of the errors experienced in Figure 4.7, a comparison of the model data to the SimSat gyro data is not expected to offer much insight. Figure 4.9 shows two identical tests in the pitch rotational direction with the roll and yaw data removed.

The pitch data as shown in Figure 4.9 confirms that the mechanical gyro data for pitch does not give enough accuracy for the attitude model to track for comparative reasons. The 20 second pitch test on the left in Figure 4.9 was the only test in about 10 total pitch tests that showed any promise of a good gyro model combination. The model seems to track the mechanical gyro with a slight lag. It is one of a few tests which caught the pitch angle switching directions by the change in slope at about the 8 second mark.

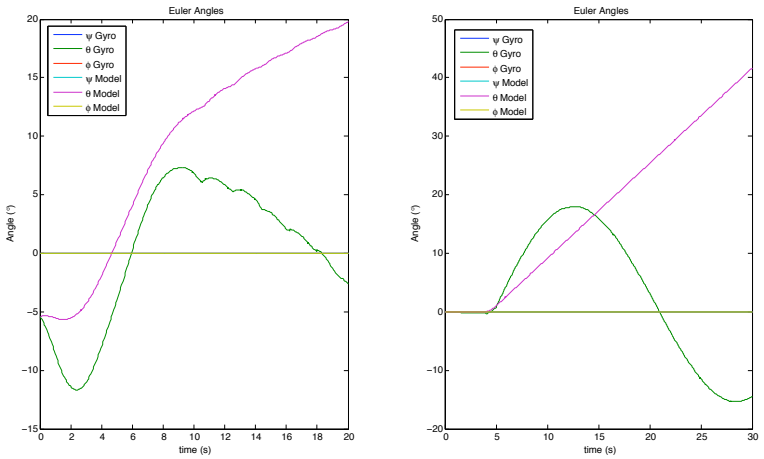

Figure 4.9: Two Pitch Test Runs

The majority of the tests looked like the plot on the right of Figure 4.9. The model data on the right looks like there is a constant torque producing a constant rate change in pitch angle with no end in sight. It looks like there may be some correlation between the model and gyro data in the first eight or so seconds. Figure 4.10 shows the telemtry data (reaction wheel rate in rpm and angle from mechanical gyroscope in radians) from SimSat.

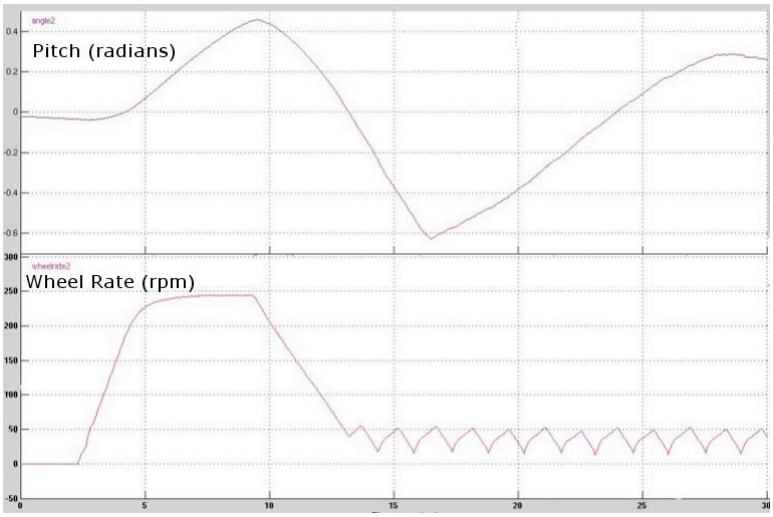

Figure 4.10: SimSat's Reaction Wheel and Mechanical Gyro Pitch Telemetry Data

The telemetry data of the reaction wheel for the pitch tests, as sampled in Figure 4.10, indicate a quick ramp up in speed followed by a constant spin rate. Then, as in those tests with a profile similar to the graphs in Figure 4.10, the reaction wheel spins down to a rate nearing zero, but at a much slower rate than the initial movement. This is seen in the left graph in Figure 4.9 by the change in slope. The MOI of the reaction wheel was changed to 4 and 4.75 lbft^2 on the right and left graphs of Figure 4.11 respectively.

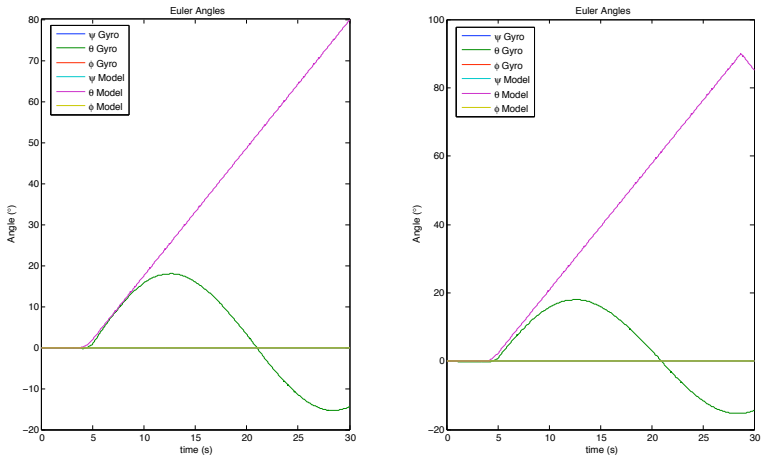

Figure 4.11: Pitch Test with 4lbft^2(left) and 4.75lbft^2(right) Reaction Wheel MOI

The increase in reaction wheel MOI suggests that there is a similar error in the MOI model of SimSat for pitch on the same order of magnitude as yaw. This suggests that the program for estimating SimSat's MOI properties is off in at least two directions, possibly due to mechanical gyro drifting and accuracy errors.

The benefit of being in a lab environment with a physical model is that the data can be compared to what was occurring visually. The pitch test made SimSat rock back and forth in the pitch axis, much like is shown in the mechanical gyro data of Figure 4.9. This seems to indicate that gyro drift and accuracy are not affecting the data as much as expected in these particular tests. As indicated in the graphs, however, none of the reaction wheel telemetry data is indicating a change in direction from the reaction wheels. An un-modeled torque, such as gravity, (due to

the flexibility of SimSat's structure as previously discussed) is helping SimSat return to its starting position is most likely the cause of this behavior. More investigation needs to be done in order to determine the cause of this behavior. The data from the pitch testing illustrates the importance of having and understanding an accurate model of the satellite and its surrounding environment. Without knowing all the forces acting on the satellite, determining satellite attitude from attitude controllers is not very reliable.

4.3.3 Roll. With the yaw and pitch rotational directions completed, there is only the roll, ϕ, direction left to discuss. The roll direction was thought to be the most impacted by gravity due to Simsat wanting to return to an equilibrium position. The two pitch tests shown in Figure 4.12 surprisingly look like the best data collected thus far.

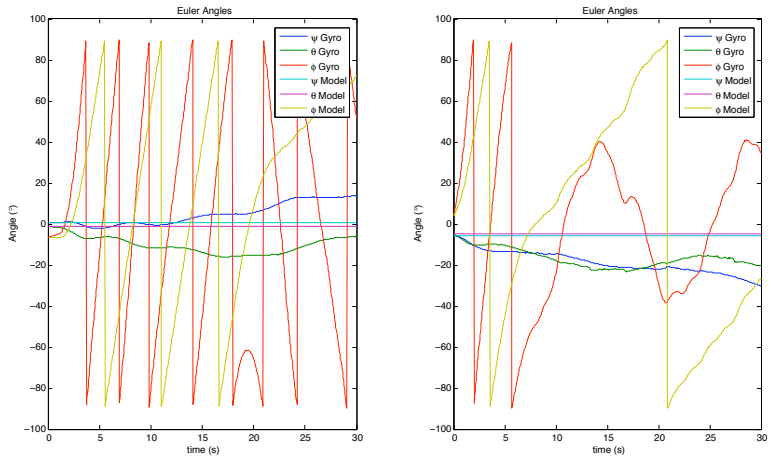

Figure 4.12: Two Roll Test Runs

The pitch and yaw mechanical gyros seem to drift throughout the testing with the yaw gyro, ψ, seeing a 30° change in orientation in the left graph of Figure 4.12. Meanwhile, the pitch, θ, shows up to a 20° drift in the thirty second test. These drift rates well exceed those found in the two minute mechanical gyro drift tests that were conducted, indicating that gyro drift may be more of a problem while in SimSat is in motion.

The data seems to mimic the gyro and what was seen in the lab except for a shift in the data which indicates a similar MOI problem as seen in the yaw and pitch directions. The MOI of the reaction wheel was changed to 4 and 4.75 lbft^2 on the right and left graphs of Figure 4.13 respectively.

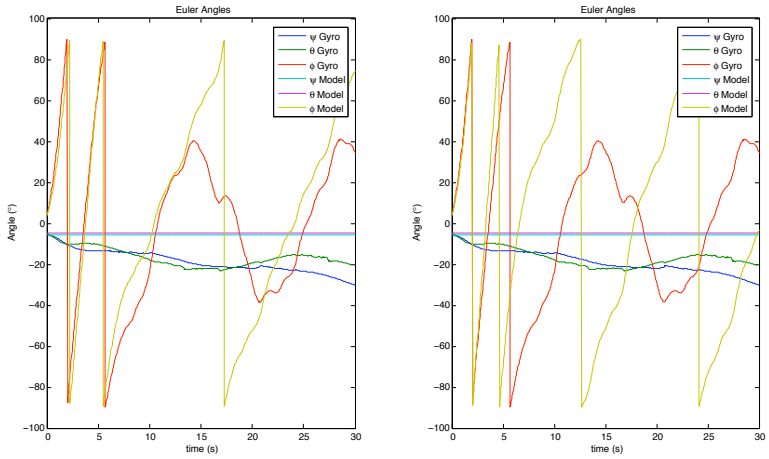

Figure 4.13: Roll Test with 4lbft^2(left) and 4.75lbft^2(right) Reaction Wheel MOI

With the reaction wheel MOI set to 4 lbft^2, the attitude determination model and the gyro data are aligned for the first 15 seconds of testing. After this point, it seems that gyro drift and the effects of gravity start to affect the results as the reaction wheels slow down. While the reaction wheel MOI of 4.75 lbft^2 appears to be too high, 4 lbft^2 is just about right and about the same magnitude of the gain needed in the yaw and pitch tests. The similarity of a gain of about 2 to the reaction wheel MOI in all three axes indicate that the error may lie within the reaction wheel MOI. If SimSat's MOI was the problem, a gain in the roll direction would be significantly lower since the roll MOI is an order of magnitude lower than the pitch and yaw directions which are very similar. This is because of the ratio of the MOI of the reaction wheel to the addition of SimSat's and the reaction wheel's MOIs as shown in Equation 2.76. The cause is not apparent and more investigation is required to determine the cause of this discrepancy. MOI data for both the reaction wheel and SimSat fell in line with MOI data used in previous theses concerning SimSat.

4.3.4 Three Degrees of Freedom. Knowing how the attitude controller-based determination model behaves in each of the three body axes separately, a three rotational degree of freedom test was conducted. From the results in sections above, it is assumed that model would not be easily compared to the orientation data provided by the mechanical gyro. Figure 4.14 shows the quaternion and Euler-angle representations of a test in which all three axes are actively rotating.

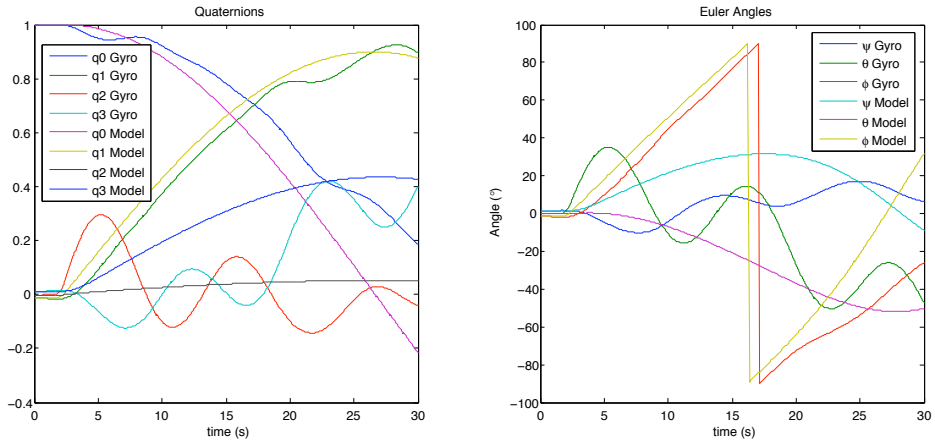

Figure 4.14: Three Rotational Degrees of Freedom Test

The test results shown in Figure 4.14 uses the calculated reaction wheel MOI of 2.08 $\text{lb}ft^2$. The model tracks better than expected. The pitch, θ, gyro data still shows the rotation of that direction and continues past the $-40°$ mark as expected. The combination of the roll and the pitch and the yaw allows SimSat to reach an orientation that allows the pitch angle to rotate past its 30° limit. This is not obvious looking at the Euler-angle representations, and is better seen in the quaternion representation of orientation.

The q_0 in Figure 4.14, shows that the attitude determination model of SimSat is rotating at approximately the same angles as the gyro data. However, the difference between the model and gyro data comes from the directions in which SimSat is turning. The q_1 data for both the model and the gyro align except for the small time delay between the two as is seen in the euler-angle representation. The roll data, ϕ,

from the model seems to lead the roll gyro data by about a second, but other than that, it nicely aligns with the gyro data throughout the thirty second test in which all three reaction wheels are spinning as shown in the Euler-angle plot on the left of Figure 4.14. This is the best indicator that a attitude controller-based determination model is feasible in tracking the orientation of a satellite. However, much work still needs to be done to increase the fidelity of the model.

The q_2 data represents the rotation about the pitch axis. It shows that the mechanical gyro is changing in direction while model continues changing in the same direction. This is the same problem as shown in the roll tests above. Also, it can be seen that as the yaw axis moves in line with the initial pitch axis and the pitch moves in line with the initial yaw axis, that the traits of the lines switch. The yaw data shows more of an oscillation towards the end of the test as the pitch data shows the oscillation dampening out some. This could be an indicator of an external torque, such as air current or gravity having an effect on the model, preventing better comparisons between the mechanical gyro data and the attitude determination from being made.

4.3.5 Summary. In summary, this chapter discussed the results from the attitude controller-based determination model simulation and experimental tests that were completed with SimSat. The first main point made was that in order to increase accuracy and reduce error of a controller-based attitude determination system the satellite's system and surroundings need to be well known, understood, and built into the model. This point was first seen in the simulation from Capt French that was conducted, and then reinforced by the data in the pitch axis. By moving to a fiber-optic gyro, the gyro drift rates on SimSat showed significant improvement in comparison with the mechanical gyro used during testing. New gyros should have a positive impact on future testing once fully-integrated with SimSat. The useable test time for comparisons is cut down to about 10-15 seconds due to the extremely poor drift rates associated with the mechanical gyros. This negatively

impacts the results of the tests conducted. One last point worth noting is that using attitude controllers for attitude determination requires another system for determining attitude with respect to some reference orientation. Finally, it can be shown from the test results in this chapter that attitude determination from telemetry data acquired from attitude controllers is at the very least feasible, but high fidelity models are needed in achieving the accuracy needed for health monitoring systems and for models and simulations used in design and academic research. Internal and external disturbances need to be accounted for in the model, as well as flexible body effects. The work herein represents a baseline to start increasing model fidelity.

V. Conclusions and Recommendations

5.1 Conclusions

Potential uses for determining attitude determination from telemetry data provided by the attitude controllers that can be found on spacecraft today are in the areas of satellite health monitoring systems and modeling and simulations. Existing health monitoring programs can benefit from using additional data to determine if a satellite is working as it should. Current models of the space environment and the satellite's themselves can be improved by comparing data from a satellite's gyroscopes to an attitude determined from its controller signals. Improved models mean better results in the research, design, and operational phases of a satellites life; all of which have cost savings associated with them. Attitude determination can even be used as a last resort in case the main attitude determination should go offline.

In order to get realistic test results, good equipment must be used. New gyros were purchased some time ago for SimSat, but it is just recently that they have been able to communicate with SimSat. The gyro drift rates for both the mechanical gyro used for testing and the new gyro recently installed on SimSat were captured showing that there is a significant improvement in gyro drift errors by replacing the mechanical gyros with to a fiber-optic ones. Even though these experiments were not able to benefit from the new gyros, future research projects will profit from the more consistent and reliable data provided by the fiber-optic gyros. However, there is still work needed to be done to ensure that the gyros are working properly and that they are well integrated with SimSat.

An attitude controller-based determination model was created and tested through software simulations and through hardware testing of AFIT's Simulated Satellite, SimSat. The software simulation verified the baseline model created for using a satellite's attitude controllers' telemetry data for its attitude determination. Tests involving SimSat proved that it is feasible to determine satellite orientation from

attitude controllers, but a lot of work remains to be done in improving the fidelity of the model in order to achieve more accurate results.

The three main points that came to light from testing the attitude controller-based model was:

- The controller systems need to be well-known and understood so that accurate models can be built.
- If not using the model to detect unknown forces, then external forces need to be well known to track attitude with any kind of precision.
- Reference orientation must be obtained to update the attitude controller-based determination model to prevent errors of drifting from un-modeled forces.

Now that testing has shown that it is feasible to track attitude orientation from controller information over short periods of time, more research needs to be done in implementing ways to help make advances in satellite health monitoring systems and to better understand the forces that a satellite encounters in space. This latter point is especially critical when trying to determine the dynamical systems involved when changing the configuration of a satellite. Models such as Capt Dabrowski's MOI Estimation Algorithm can be used in conjunction with the attitude controller-based determination model to figure out how a satellite's MOI changes and how it affects the dynamics of a spacecraft while changing its configuration, such as when it deploys its solar arrays.

5.2 Recommendations for Future Study

Even though testing didn't show results as accurate as one would like, the Attitude Controller-Based Determination Model that was created is a great place to begin new research. This research topic is untapped and has plenty of academic and commercial value associated with it. Below is a list of things that should be considered for future research.

- Investigate the causes for an apparent gain between SimSat and the Attitude Controller-Based Determination Model that was created.
- Attempt to repeat data spikes in the new fiber optic gyros on SimSat that were seen in analog testing of the gyros off of SimSat. Implement a low-pass filter if need be.
- Rerun tests with the newer fiber-optic gyros and compare to mechanical gyro results presented herein
- Identify the un-modeled forces on SimSat to improve the discrepancies between the SimSat software model simulations and SimSat experimental test results.
- Create algorithms to use attitude controller telemetry data to get the most valuable information for health monitoring systems or for improving the modeling of unknown forces in space.
- Update SimSat's ground station's computer software to the most recent Matlab® and Simulink® versions for students' ease of use.
- Replace RealMotionPC3D with the VR toolbox for Matlab®. This toolbox can be used by both hardware and software simulations for visualization purposes.

Appendix A. MOI Estimation Procedure

The following is taken from Appendix B: Calculation of SIMSAT Moments of Inertia, as found in Capt Kimsal's thesis [11]. The basis for this code is derived from work done by Capt Dabrowski in his thesis [3]. the following procedure put together by Capt Kimsal was used in the determination of of SimSat's principal MOI's:

> The moments of inertia of the SimSat must be correctly calculated in order for both accurate modeling in the Simulink portion of the experiment, as well as for accurate calculations when designing the controller. With reconfiguration of the SimSat between major experiment topics comes the need to recalculate its moments of inertia. A concise manner in which to perform this calculation has been created in the form of a ControlDesk experiment. The experiment, titled, "MOI_test.cdx" must be initially loaded through the dSPACE® ControlDesk software (the reader is assumed to have a basic knowledge of both ControlDesk and Matlab®). After the moi_test.ppc file has been loaded to the ds1005 platform aboard SimSat, the user need only activate the Animation mode and the experiment begins automatically. The experiment is designed to actuate one reaction wheel during each run. The reaction wheel, is accelerated to 250 rad/sec, and the resulting spacecraft inertial angles are recorded. The test does need to be reconfigured in order to test all three reaction wheels. Two steps are required to accomplish this: 1) in the Simulink® model, "MOI_test.mdl", the step input needs to be changed to the appropriate direction, and 2) the corresponding output variable in ControlDesk needs to be linked as the recorded variable. After the completion of the data gathering, the data must be saved. It is saved by default as a Matlab® MAT-file. After data from all directions has been gathered, the Matlab ®file "moi_test.m" can be used to determine the MOI.
>
> The file must be opened and changed to load the appropriate data files that were saved. A simple name change will accomplish this. The data is manipulated in the following fashion. A time vector is extracted from one of the data sets (they are all identical). Each data set is then parsed to extract the recorded inertial angular movements in the appropriate direction. The data is then stepped in 5% increments to determine the slope along the entire curve of angular displacement v. time. The maximum slope is used as the slope of record. If desired, the user can take data sets in both a positive and negative direction and average the

two slope results. The results of the slope are then used to solve Equation 3.1 where ω I has been fixed to 250 rad/sec for this experiment, and I_{rw} was established as 0.01955 kgm^2 during the construction of SimSat. If the user wishes to change this value, it can be changed in the Simulink model.

```
% SIMSAT Moment of Inertia Test
% Author (Dabrowski)
% Edited by Capt Matt Kimsal 11 Jan 04

% ************************************************************
% NOTES:
%   This code is used to determine the baseline MOI matrix for
%   the SIMSAT.
%   It assumes a rigid body and that the reaction wheels are
%   aligned with the principle axes (SIMSAT doesn't quite
%   match that, but it's close).
%
%   This code will produce the baseline MOI if it is fed in
%   the data files containing time-stamped histories of the
%   angular displacements in each nominal direction (roll,
%   pitch, yaw).  These data files can be obtained by running
%   the dSPACE experiment 'MOI_test' (assumed written by
%   Dabrowski).  The model is already set up to accomodate
%   this particular file, so the only action necessary is to
%   load it on the ds1005 aboard SIMSAT, and start the
%   Animation mode (sorry, you'll have to learn
%   ControlDesk on your own).  The data capture will start
%   automatically and you just have to save the data after
%   the 20-second capture is complete.  Hope this helps.
%   MBK 1/11/04
% ************************************************************

clear
format long e
clc

%SHAFT (From Motor Manual)
I_shaft=2.12e-4;

%DISK
```

```
r_disk_min=.375/2 * .0254;
r_disk_max=(8.625-2*.375)/2 * .0254;
L_disk=.25 * .0254;
d_disk=2700;
I_disk=.5*pi*d_disk*L_disk*(r_disk_max^4-r_disk_min^4);
I_disk_lbft2=I_disk*23.730360404;

%HOOP
r_hoop_min=(8.625-2*.375)/2 * .0254;
r_hoop_max=8.625/2 * .0254;
L_hoop=1.1875 * .0254;
d_hoop=8000;
I_hoop=.5*pi*d_hoop*L_hoop*(r_hoop_max^4-r_hoop_min^4);
I_hoop_lbft2=I_hoop*23.730360404;

%TOTAL
I_wheel=I_shaft+I_disk+I_hoop;
I_wheel_lbft2=I_wheel*23.730360404

%CHECK THAT MASS IS APPROX 2.040, 2.070 KG
m_disk=d_disk*pi*(r_disk_max^2-r_disk_min^2)*L_disk;
m_disk_lb=m_disk*2.204622622;
m_hoop=d_hoop*pi*(r_hoop_max^2-r_hoop_min^2)*L_hoop;
m_hoop_lb=m_hoop*2.204622622;
m_wheel=m_disk+m_hoop;
m_wheel_lb=m_wheel*2.204622622;

%USE WHEEL MOI & DATA TO GET SIMSAT MOI

load yaw_pos_12jan04
load yaw_neg_12jan04
load roll_pos_12jan04
load roll_neg_12jan04
load pitch_pos_14jan04
load pitch_neg_14jan04

t = yaw_pos_12jan04.X.Data;
length_t = length(t);

yp = yaw_pos_12jan04.Y.Data;
yn = yaw_neg_12jan04.Y.Data;
pp = pitch_pos_14jan04.Y.Data(1:120);
```

```
pn = pitch_neg_14jan04.Y.Data(1:120);
rp = roll_pos_12jan04.Y.Data;
rn = roll_neg_12jan04.Y.Data;

plot(t,yp,t,yn,t,rp,t,rn);
legend('yaw_{pos} 2 Jan','yaw_{neg} 2 Jan',
'roll_{pos} 2 Jan',
'roll_{neg} 2 Jan')

all=[yp;yn;rp;rn];
window=20;
for set=1:4
    max_slope=0;
    for start_=1:length_t-window
        finish_=start_+window;
        current_group=all(set,start_:finish_);
        current_t=t(start_:finish_);
        p = polyfit(current_t,current_group,1);
        if abs(p(1))>max_slope
            max_slope=abs(p(1));
        end
    end
    max_slope_all(set)=max_slope;
end

last_yaw_pos = yaw_pos_12jan04.Y.Data(length_t);
last_yaw_neg = yaw_neg_12jan04.Y.Data(length_t);
last_rol_pos = roll_pos_12jan04.Y.Data(length_t);
last_rol_neg = roll_neg_12jan04.Y.Data(length_t);

length_t = 120;
not = zeros([1,size(pp,2)]);
all=[not;not;not;not;pp;pn];
window=10;
for set=5:6
    max_slope=0;
    for start_=1:length_t-window
        finish_=start_+window;
        current_group=all(set,start_:finish_);
        current_t=t(start_:finish_);
        p = polyfit(current_t,current_group,1);
        if abs(p(1))>max_slope
```

```
            max_slope=abs(p(1));
        end
    end
    max_slope_all(set)=max_slope;
end

max_slope_all

last_pit_pos = pitch_pos_14jan04.Y.Data(length_t);
last_pit_neg = pitch_neg_14jan04.Y.Data(length_t);

avg_last_yaw = (abs(last_yaw_pos)+abs(last_yaw_neg))/2;
avg_last_pit = (abs(last_pit_pos)+abs(last_pit_neg))/2;
avg_last_rol = (abs(last_rol_pos)+abs(last_rol_neg))/2;

yaw_slope=(max_slope_all(1)+max_slope_all(2))/2;
pitch_slope=(max_slope_all(5)+max_slope_all(6))/2;
roll_slope=(max_slope_all(3)+max_slope_all(4))/2;

MOI_yaw=I_wheel_lbft2*(250-avg_last_yaw)/yaw_slope
MOI_pitch=I_wheel_lbft2*(250-avg_last_pit)/pitch_slope
MOI_roll=I_wheel_lbft2*(250-avg_last_rol)/roll_slope
% NOTE that this w is different.  Running the sim at w=250
% caused it to crash before 20 sec had elapsed.
```

Appendix B. FOG SimSat Simulink® Integration Code

The following figures illustrate the mathematical models developed in Simulink® to get the Fiber-Optic Gyroscope communicating with SimSat. These models are based upon a Matlab® script that allowed the gyroscope to work directly with Matlab®. Figure B.1 is a top level model that can be inserted in replace of the current gyroscope interface block. Position is found by integrating the orientation rates outputted from the model.

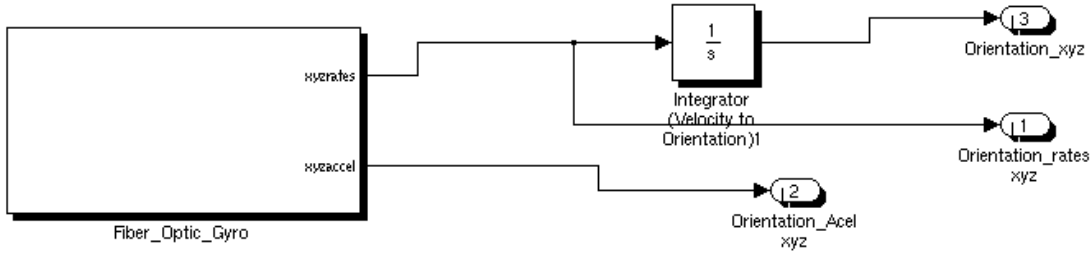

Figure B.1: Fiber-Optic Gyro - Top Level

Looking underneath the Fiber_Optic_Gyro block, Figure B.2 can be broken into four parts. The first part being the red RTW blocks which allow for communication between the gyroscope and SimSat. From those blocks, the signal is converted to a double integer and it and the width of the signal are fed into a for loop that searches for the header file and outputs a signal in the correct order. The signal then goes to a selector, from which, the correct signals are sent to the rate and accel scaling blocks where the angular velocities are outputted in radians/second and the linear accelerations are outputted in meters/second. Figures B.3, B.4, and B.5 show the models which drive the for iterator block. One thing to note is that the information from the gyro is being fed into the RTW blocks at a rate of 400 Hz in blocks of 21 bytes. The model is limited to running at a minimum time integration step of 0.05 seconds due an Autobox® limitation. Because the data is not synchronized with the

model, a search is performed on 41 bytes, or two sets of data sent, to capture the correct order of the data. This is fine as long as the time integration step is not shortened to 200 Hz. Figures B.6 and B.7 are the scaling equations that are found in the gyrorate.m. The scaling equations take the information from the signal and produce the angular rates and linear accelerations of the gyroscope.

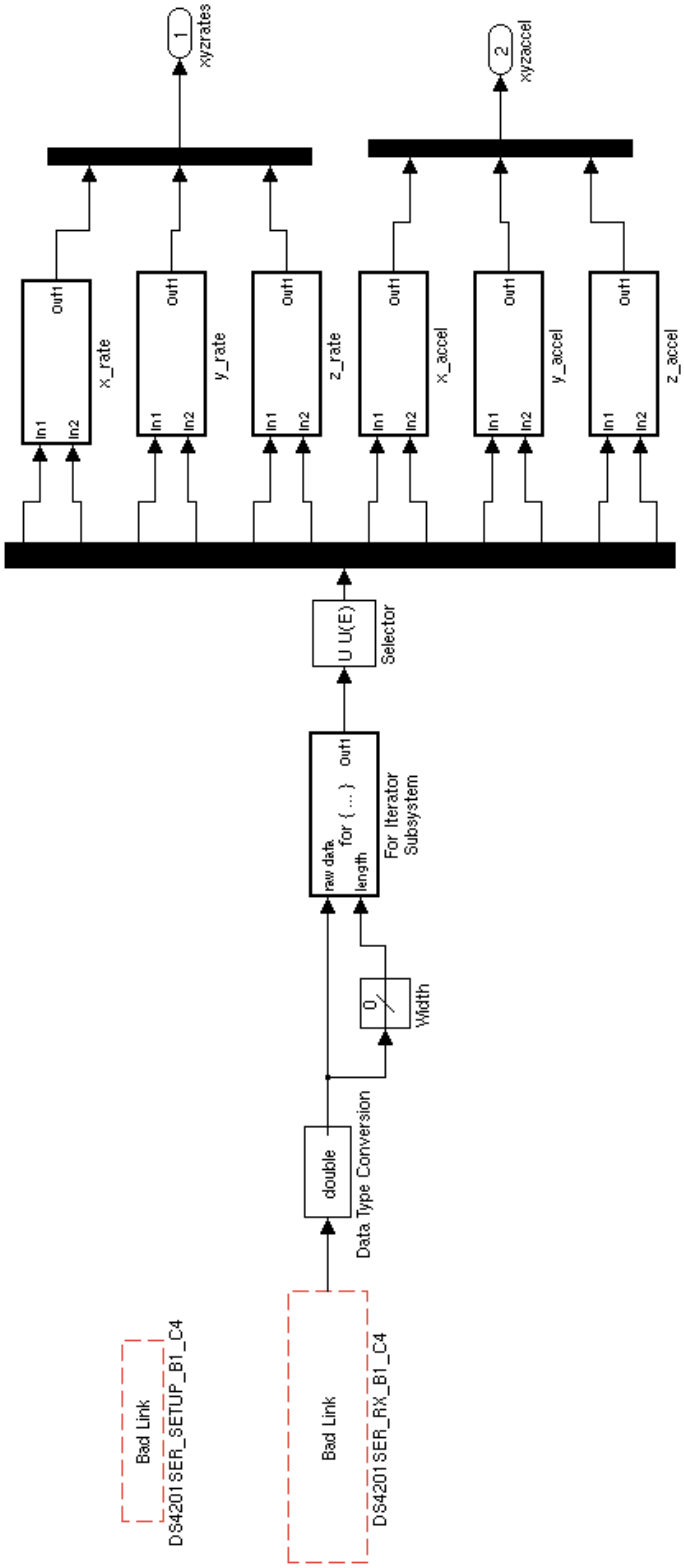

Figure B.2: Fiber_Optic_Gyro Simulink Block

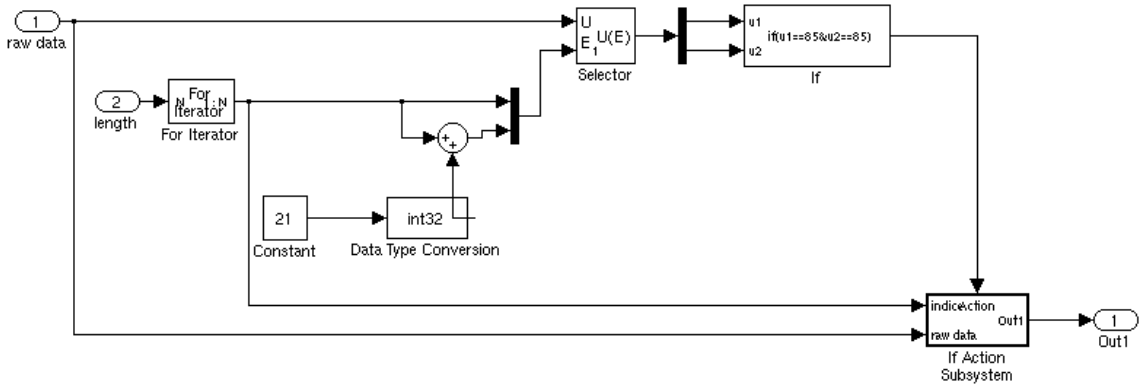

Figure B.3: LN-200 Interface Data Sync - For Iterator

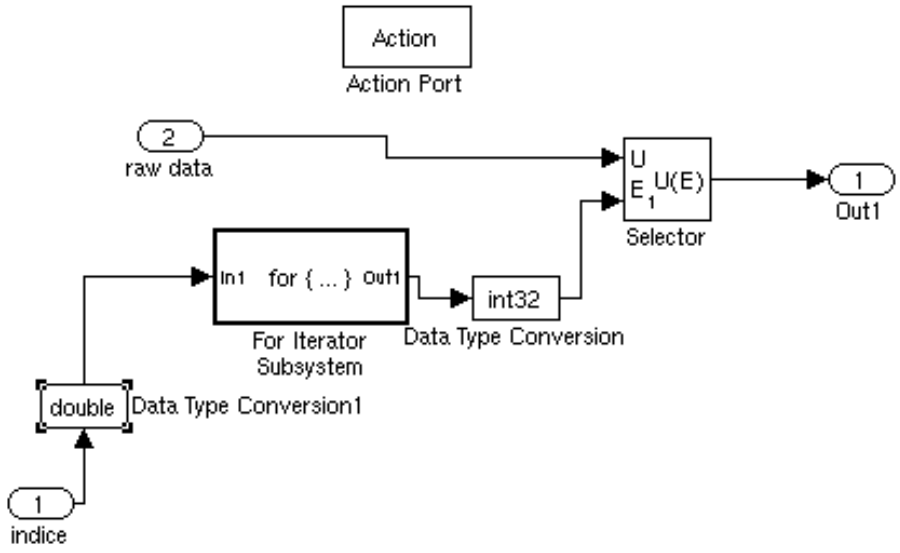

Figure B.4: Action Port to If Iterator Simulink Block

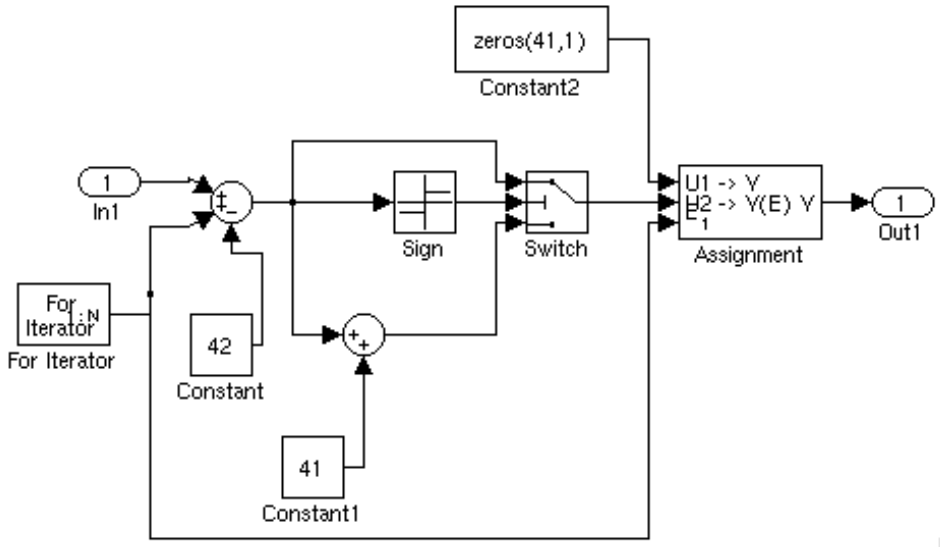

Figure B.5: Copy Data Stream into Data Vector - For Iterator

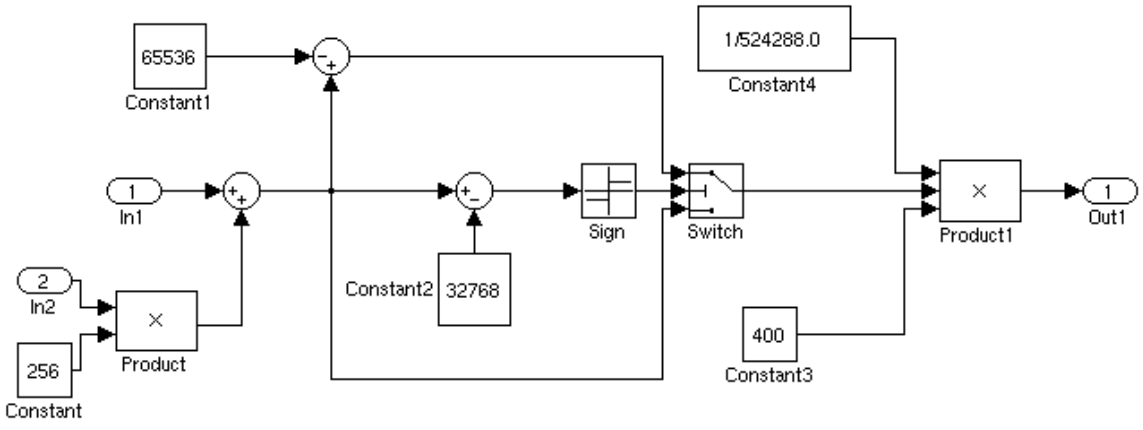

Figure B.6: Scaling Equation for Angular Rates

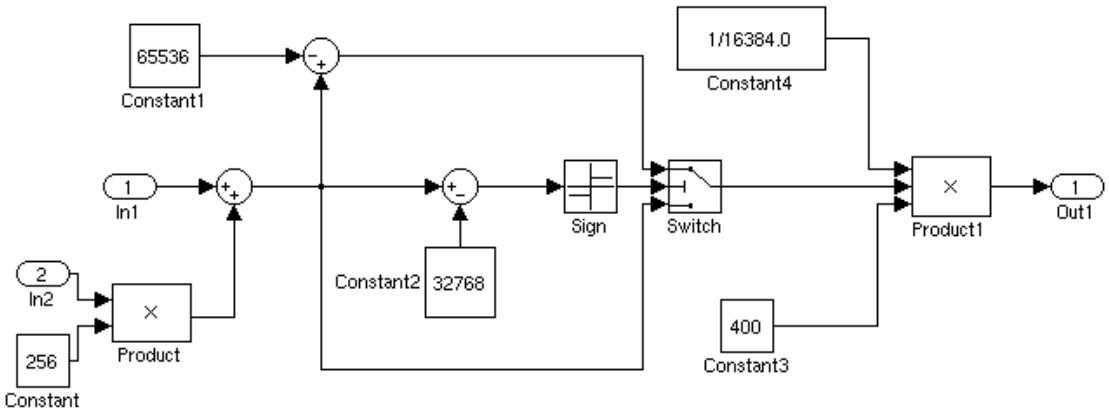

Figure B.7: Scaling Equation for Linear Acceleration

Here is the gyrorate.m filed used to scale the data received from the LN-200 board. This is the file that was used as the basis for the Simulink® blocks created to interface the the Gyro and SimSat. Outputs of this file are linear accelerations in m/s and angular rates in rads/s:

```
function [thetas]=gyrorate()
% Gyroscope Serial Input Reading
clear
port = serial('COM1','BaudRate',115200,'InputBufferSize',45,
'ReadAsyncMode','manual');
fopen(port)
tic;
lasttime=toc;
INS_FREQ = 400;  %in HZ
IMU_PK = (1.0/16384.0);
IMU_AK = (1.0/524288.0);

fprintf('Reading Data\n')
for(j=1:100)
    ordinate(j) = toc;
    data=0;
    while(data(1)~=85)
        readasync(port,45);
        data=fread(port,45);
        for(i=1:22)
            if(data(i)==85&data(i+21)==85)
                data=data(i:(i+20));
                break
            end
        end
    end
    if(data(1)~=85)
        fprintf('Data Error\n');
    end

    delta_t(j)=toc-lasttime;
    lasttime=toc;

    x_accel = (data(3)*256+data(2));
    y_accel = (data(5)*256+data(4));
```

```
z_accel = (data(7)*256+data(6));

x_rate = (data(9)*256+data(8));

y_rate = (data(11)*256+data(10));
z_rate = (data(13)*256+data(12));

if(x_accel>32768)
    x_accel = x_accel - 65536;
end
if(y_accel>32768)
    y_accel = y_accel - 65536;
end
if(z_accel>32768)
    z_accel = z_accel - 65536;
end

if(x_rate>32768)
    x_rate = x_rate - 65536;
end
if(y_rate>32768)
    y_rate = y_rate - 65536;
end
if(z_rate>32768)
    z_rate = (z_rate - 65536);
end

x_accel = IMU_PK*x_accel*INS_FREQ;
y_accel = IMU_PK*y_accel*INS_FREQ;
z_accel = IMU_PK*z_accel*INS_FREQ;

x_rate = IMU_AK*x_rate*INS_FREQ;
y_rate = IMU_AK*y_rate*INS_FREQ;
z_rate = IMU_AK*z_rate*INS_FREQ;

accel(j,1)=x_accel;
accel(j,2)=y_accel;
accel(j,3)=z_accel;
rate(j,1)=x_rate;
rate(j,2)=y_rate;
rate(j,3)=z_rate;
```

```
end
fprintf('Data Read Complete\n');
time=toc;
fprintf('Frequency was %fHz\n',100/time)
figure(1)
subplot(2,1,1)
plot(ordinate,accel(:,1),ordinate,accel(:,2),
ordinate,accel(:,3))
legend('X Acceleration','Y Acceleration',
'Z Acceleration',2)
subplot(2,1,2)
plot(ordinate,rate(:,1),ordinate,rate(:,2),
ordinate,rate(:,3))
legend('X Rate','Y Rate','Z Rate',2)

angle=rate;
angle(:,1)=rate(:,1).*delta_t';
angle(:,2)=rate(:,2).*delta_t';
angle(:,3)=rate(:,3).*delta_t';
for(i=2:length(angle))
    angle(i,:)=angle(i,:)+angle(i-1,:);
end
angle(1,:)=0;

velocity=accel;
velocity(:,1)=accel(:,1).*delta_t';
velocity(:,2)=accel(:,2).*delta_t';
velocity(:,3)=accel(:,3).*delta_t';
for(i=2:length(velocity))
    velocity(i,:)=velocity(i,:)+velocity(i-1,:);
end
velocity(1,:)=0;

position=velocity;
position(:,1)=velocity(:,1).*delta_t';
position(:,2)=velocity(:,2).*delta_t';
position(:,3)=velocity(:,3).*delta_t';
for(i=2:length(position))
    position(i,:)=position(i,:)+position(i-1,:);
end
position(1,:)=0;
```

```
% figure(2)
% plot(ordinate,angle(:,1),ordinate,angle(:,2),
% ordinate,angle(:,3))
% legend('X Angle','Y Angle','Z Angle',2)
%
% figure(3)
% plot(ordinate,velocity(:,1),ordinate,velocity(:,2))
% ordinate,velocity(:,3))
% legend('X velocity','Y velocity')%,'Z velocity',2)
%
% figure(4)
% plot(ordinate,position(:,1),ordinate,position(:,2))
% ordinate,position(:,3))
% legend('X position','Y position')%,'Z position',2)

fclose(port)
delete(port)
clear port
```

Appendix C. Simulink® Attitude Controller-Based Determination Models and Code

Following is an archive of the Simulink® models and Matlab® files that determine a satellite's attitude from reaction wheel and thruster indication telemetry data.

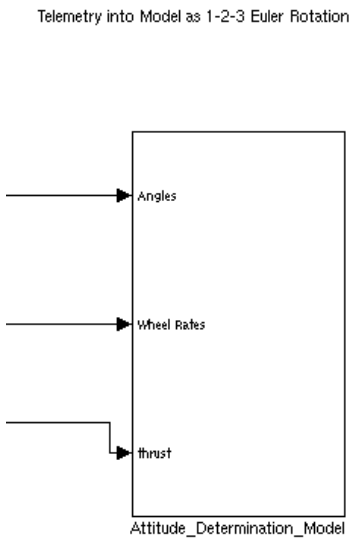

Figure C.1: Simulink Attitude Determination Block

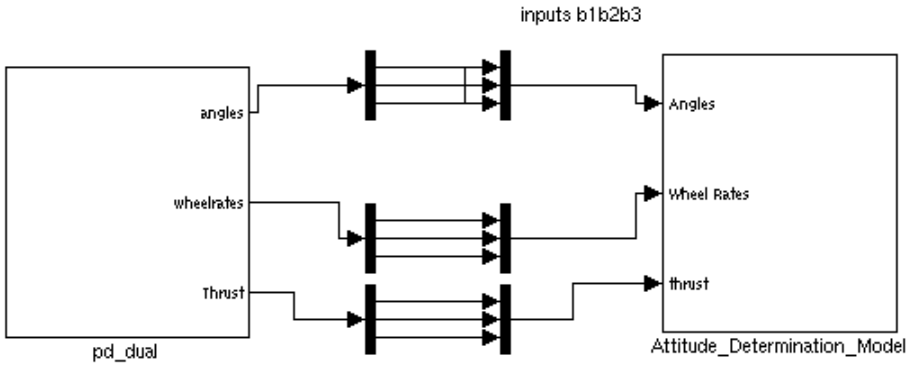

Figure C.2: Attitude Determination from PD_Dual_Sim - Top Level

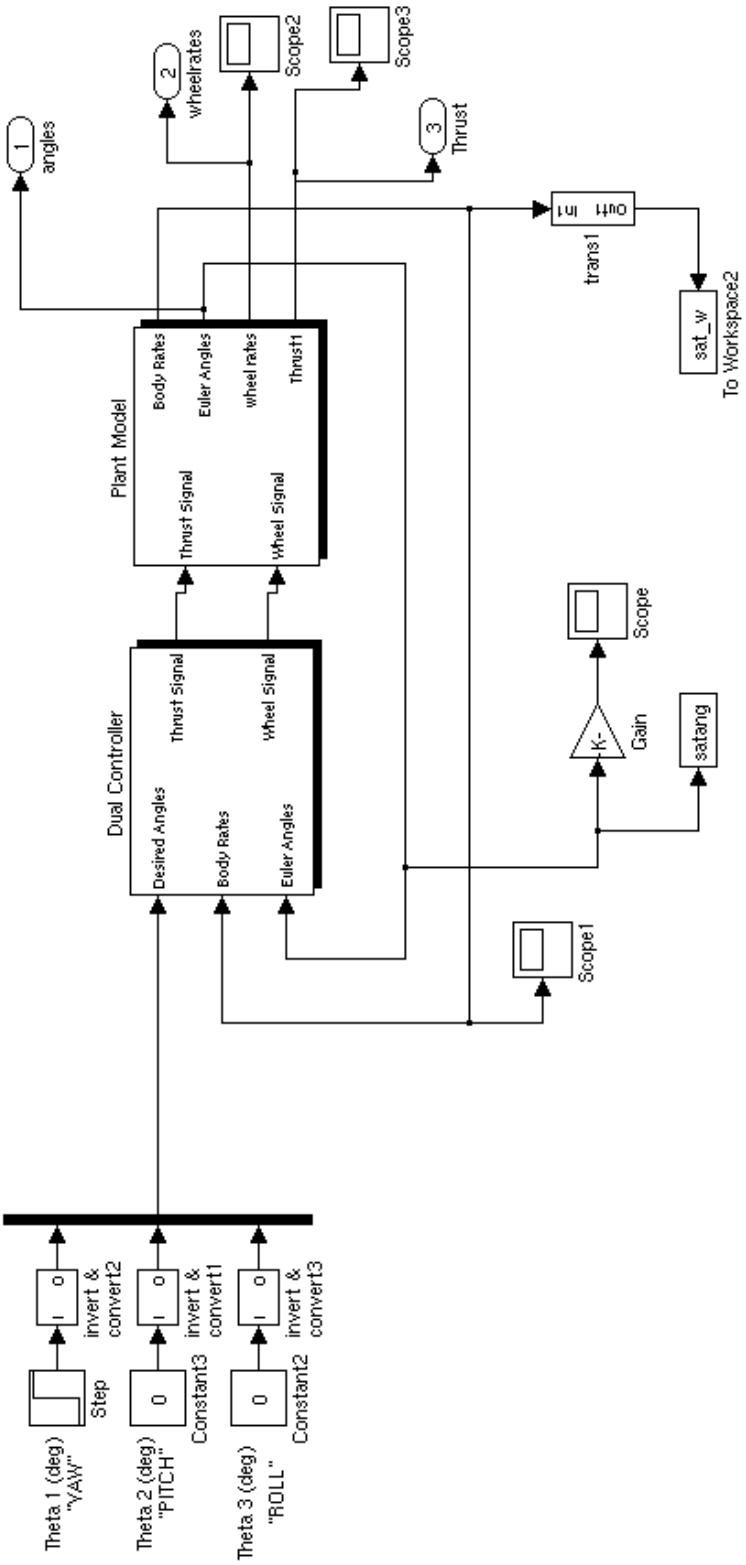

Figure C.3: PD_Dual_Sim - Top Level

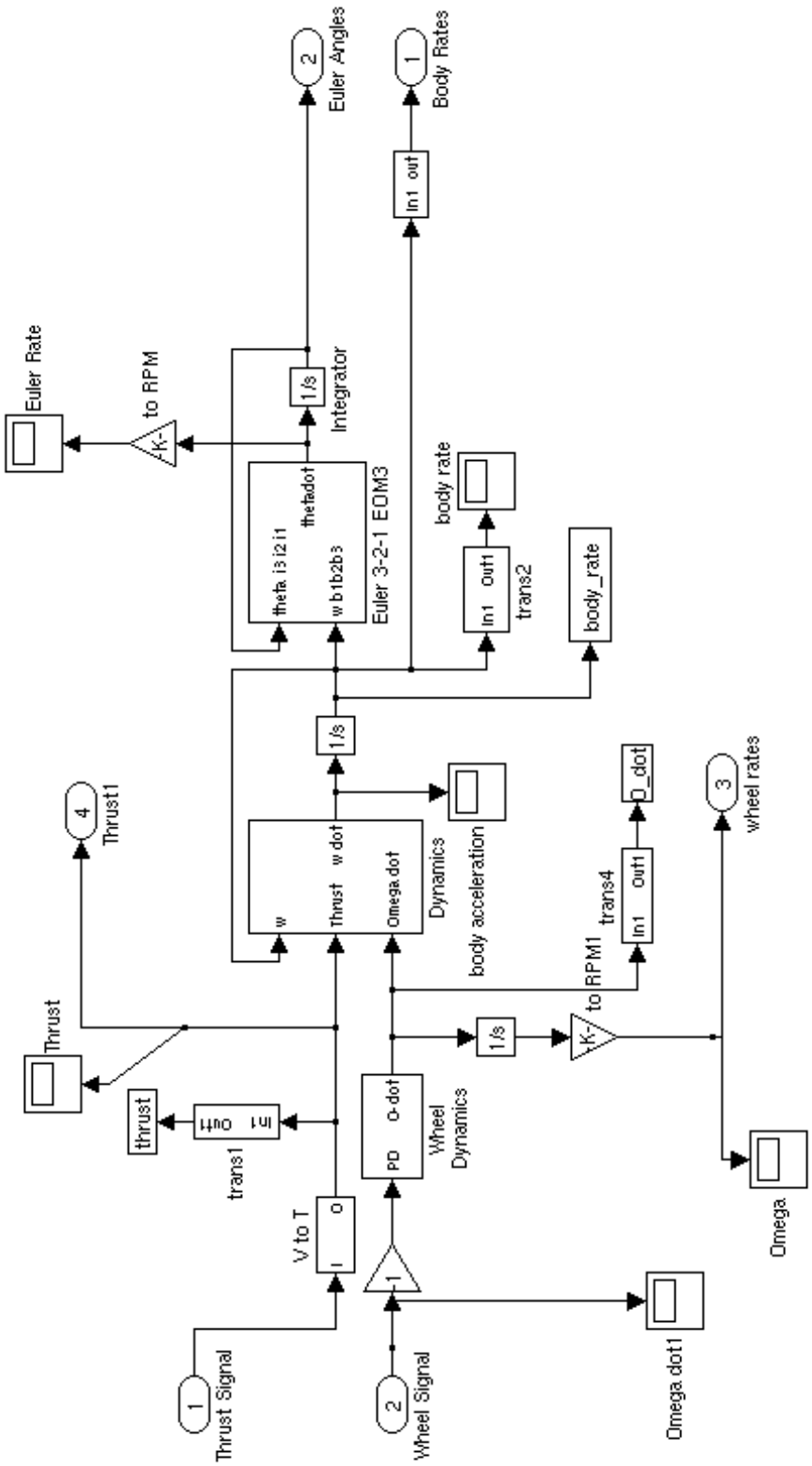

Figure C.4: PD_Dual_Sim Plant

Figure C.5: MOI Test - Top Level

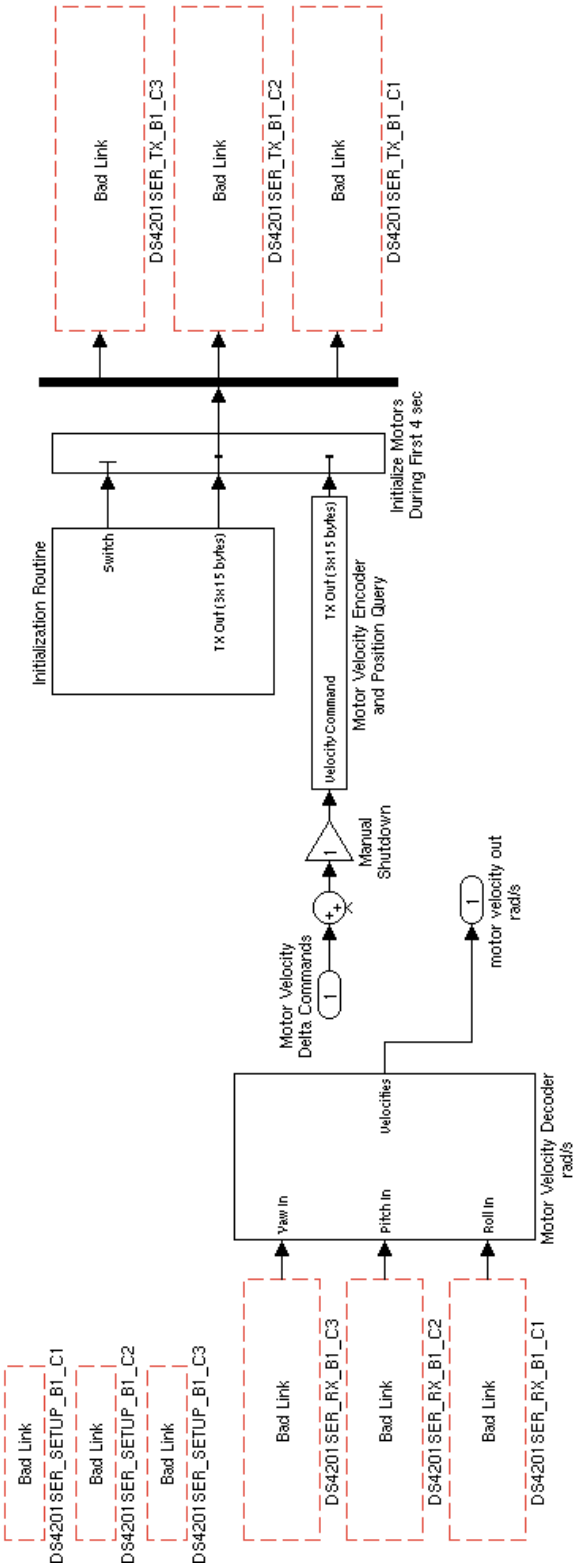

Figure C.6: MOI Test Reaction Wheel Controller

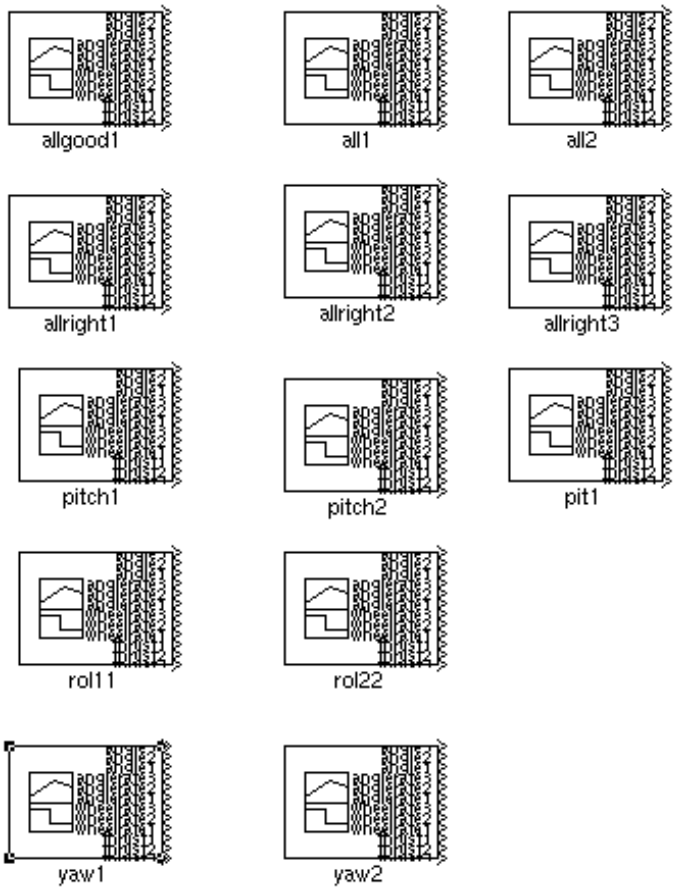

Figure C.7: SimSat Telemetry Signals

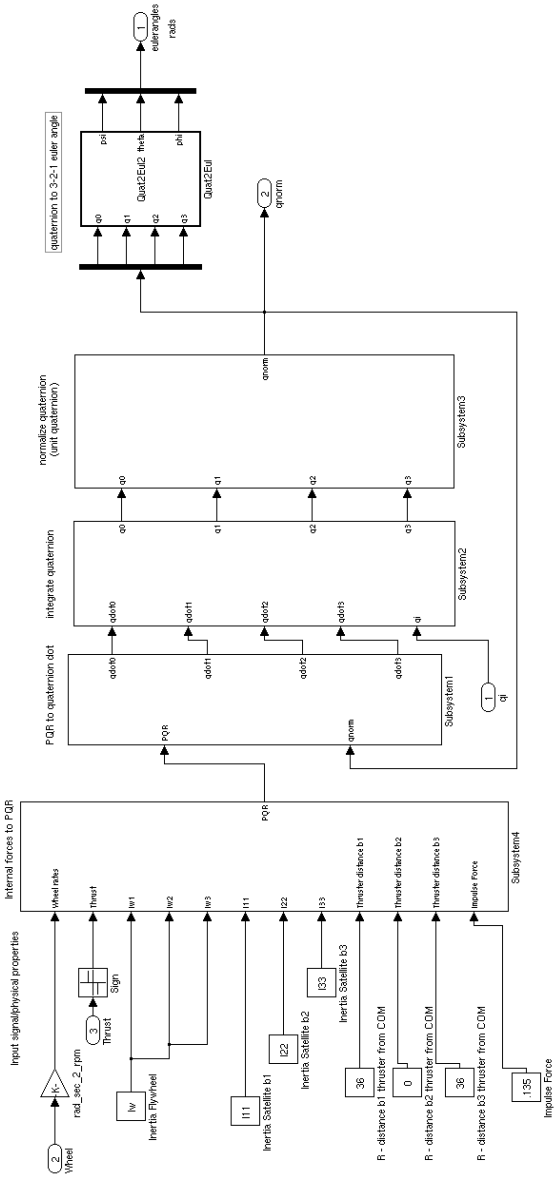

Figure C.8: Attitude Determination - Top Level

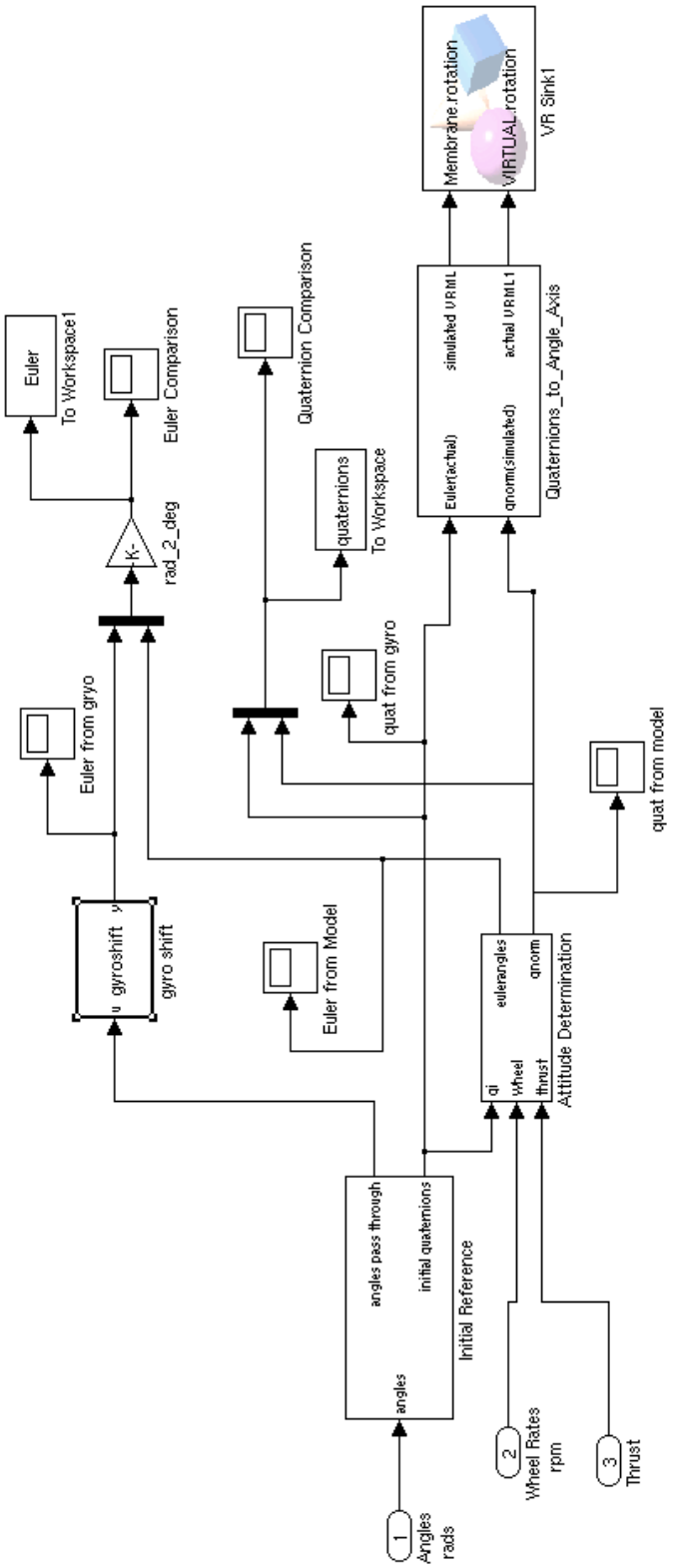

Figure C.9: Attitude Determination Model - Visual Level

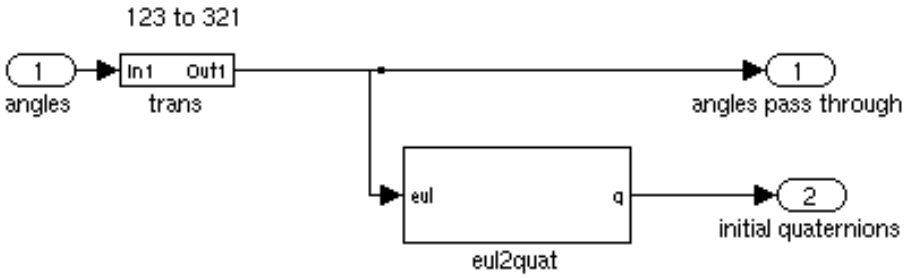

Figure C.10: Initializing Quaternion from Gyroscope

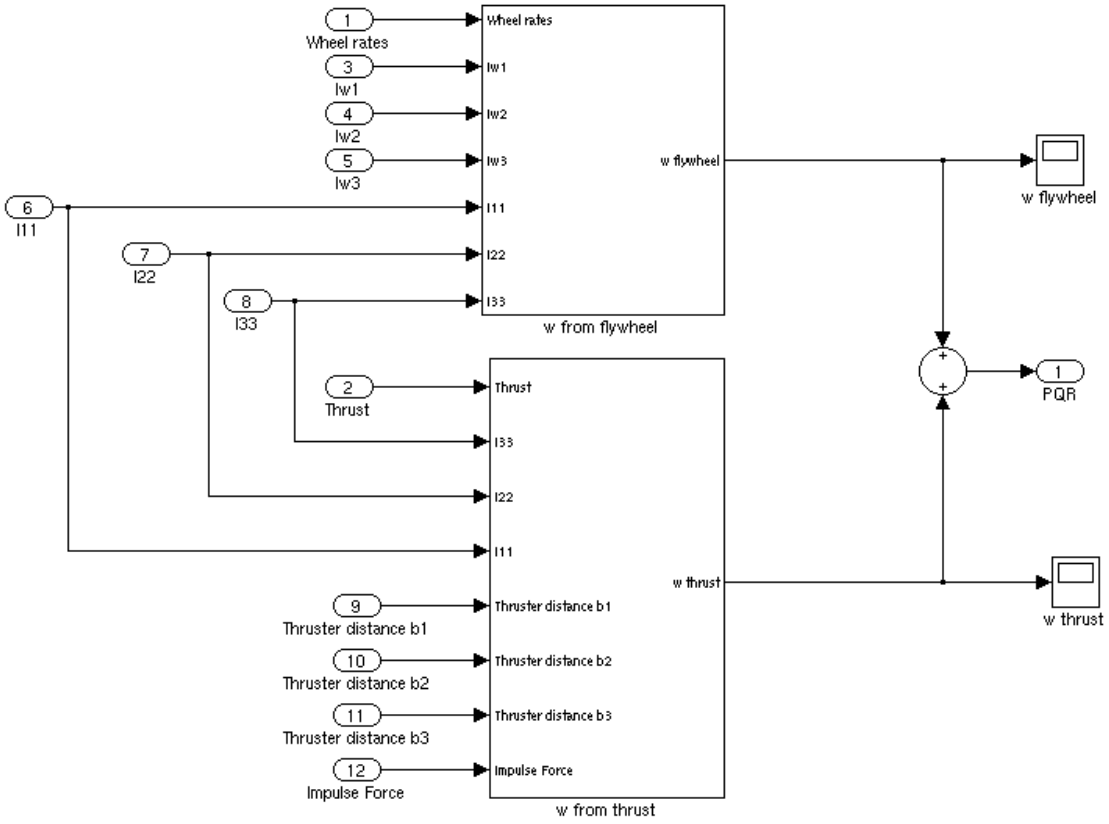

Figure C.11: Internal Forces to PQR

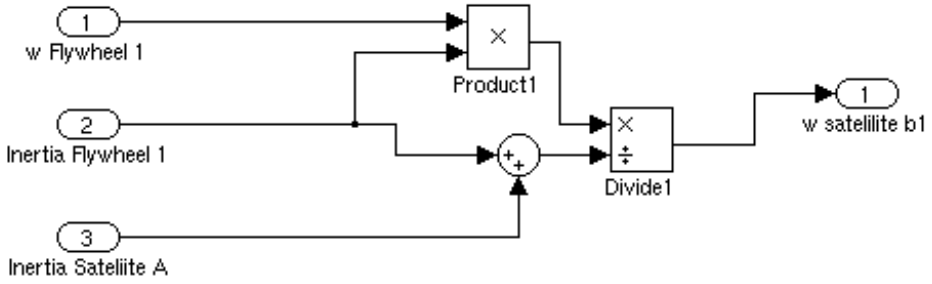

Figure C.12: Reaction Wheel Input to Satellite Rotation

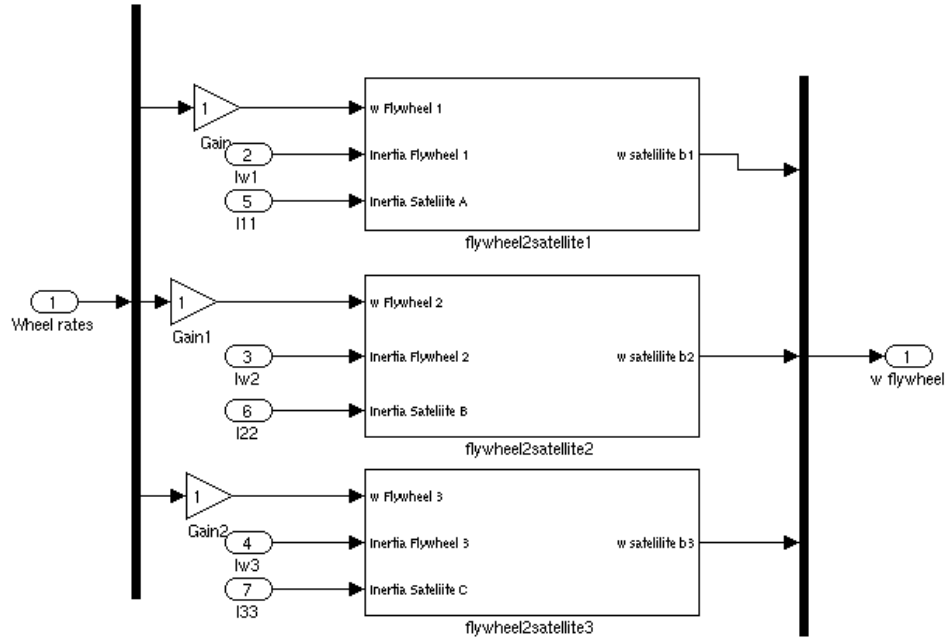

Figure C.13: Angular Rate from Reaction Wheel

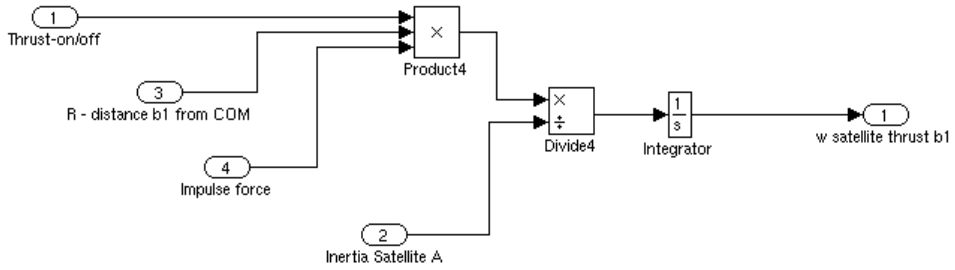

Figure C.14: Thruster to Satellite Angular Velocity

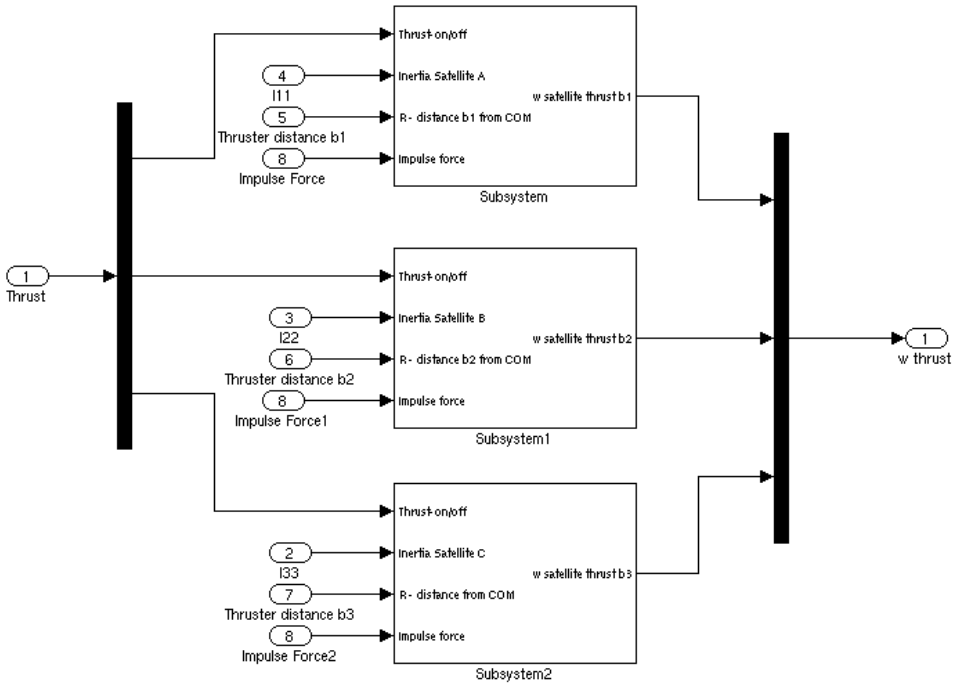

Figure C.15: Angular Rate from Thrust

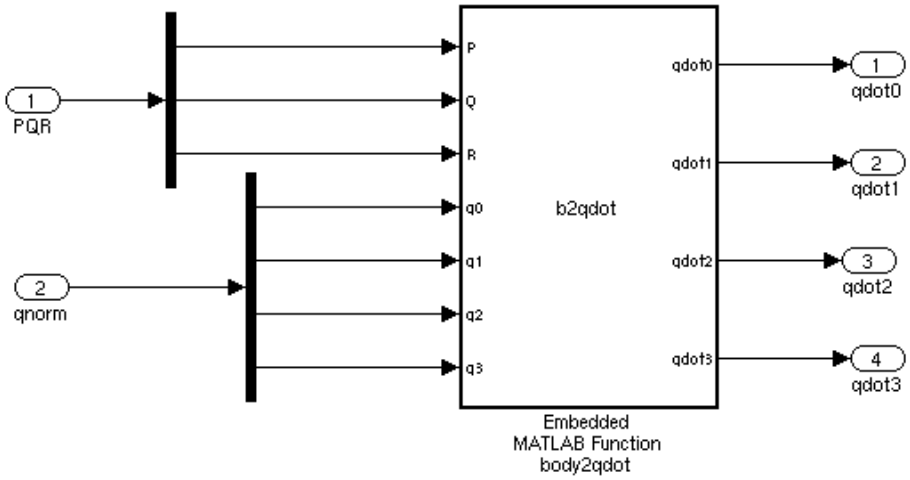

Figure C.16: PQR to Quaternion Rates

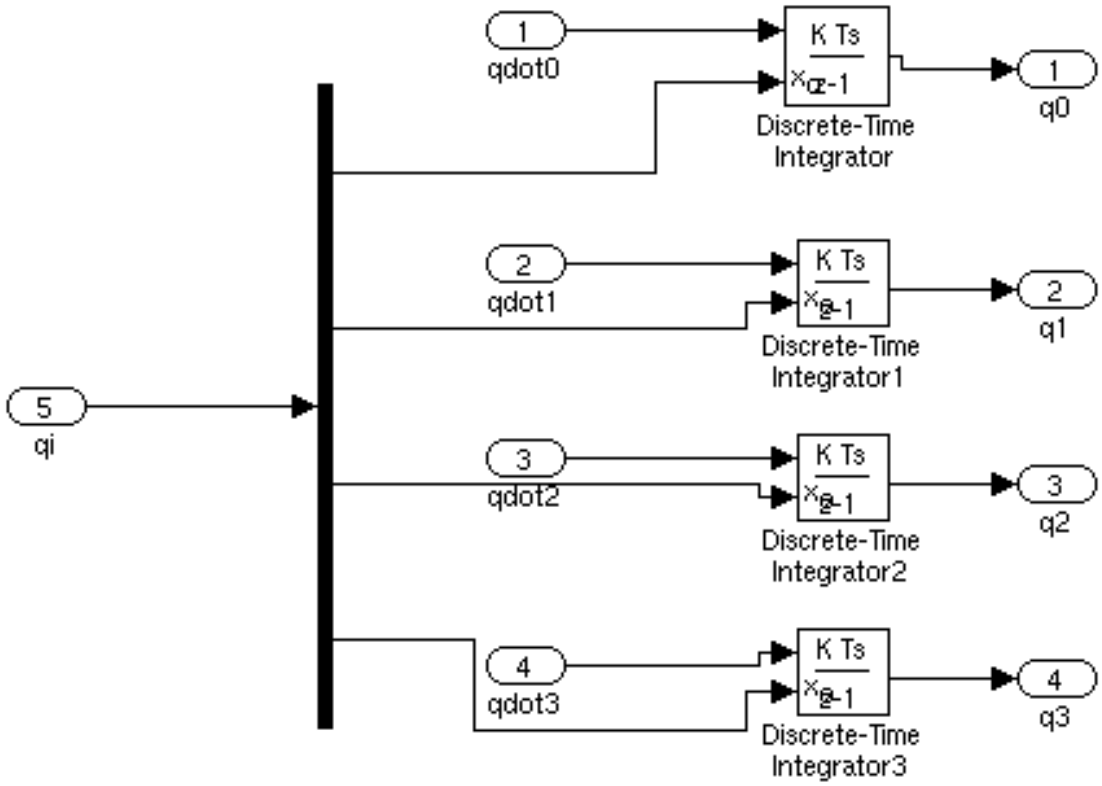

Figure C.17: Quaternion Rate Integration

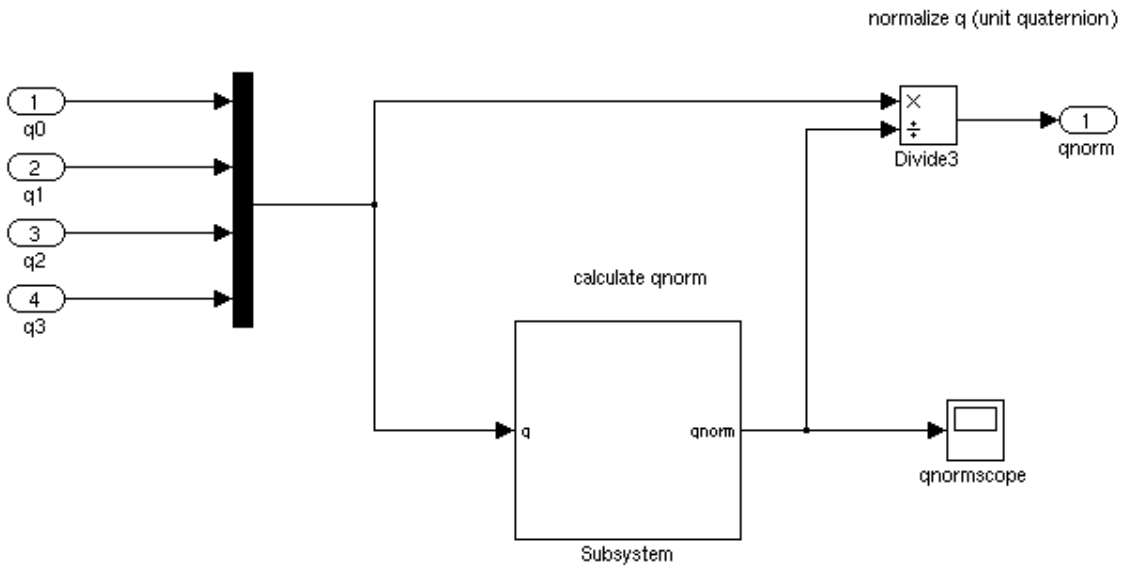

Figure C.18: Quaternion Normalization

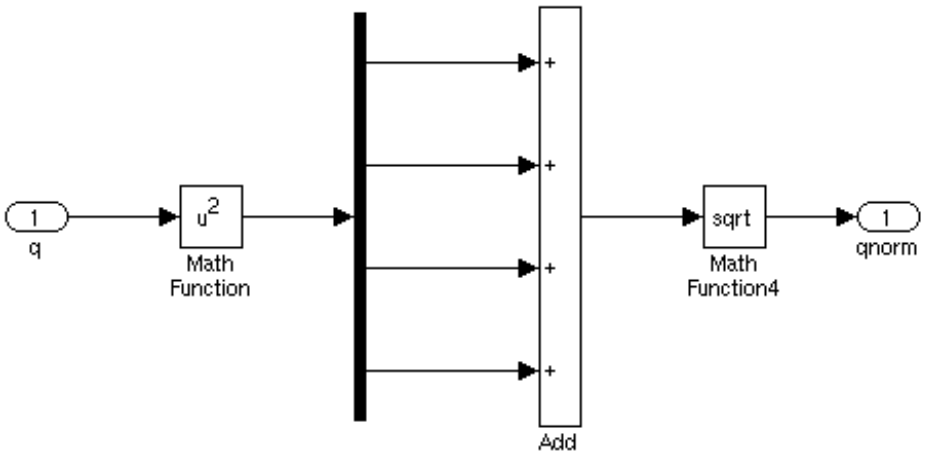

Figure C.19: Calculating Normalized Quaternion

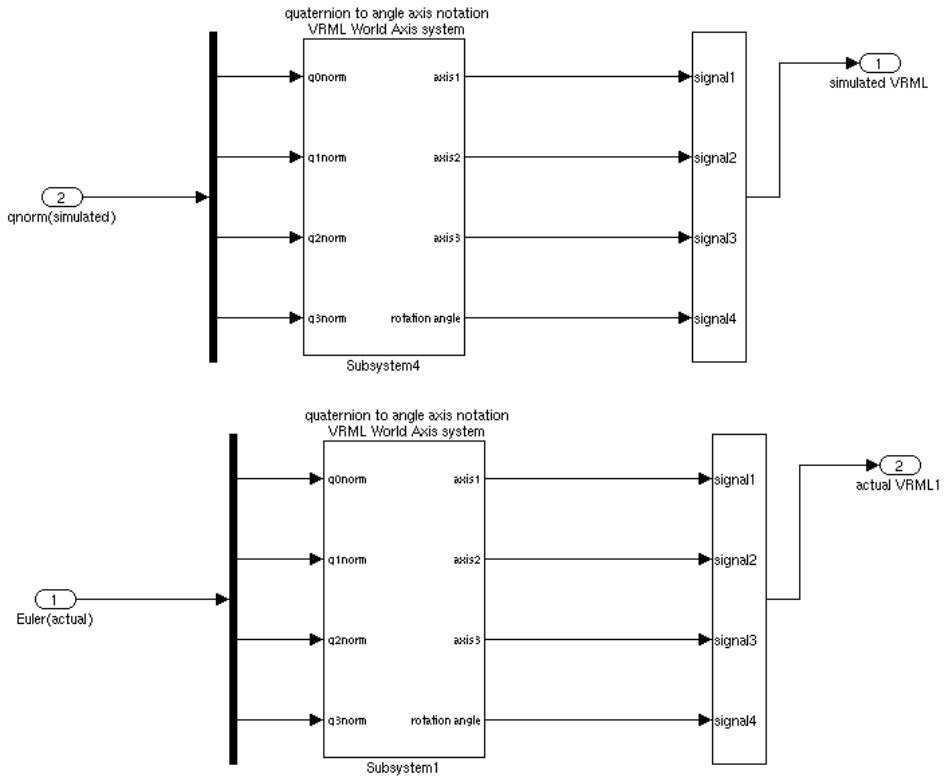

Figure C.20: Quaternion to Angle Axis Representation

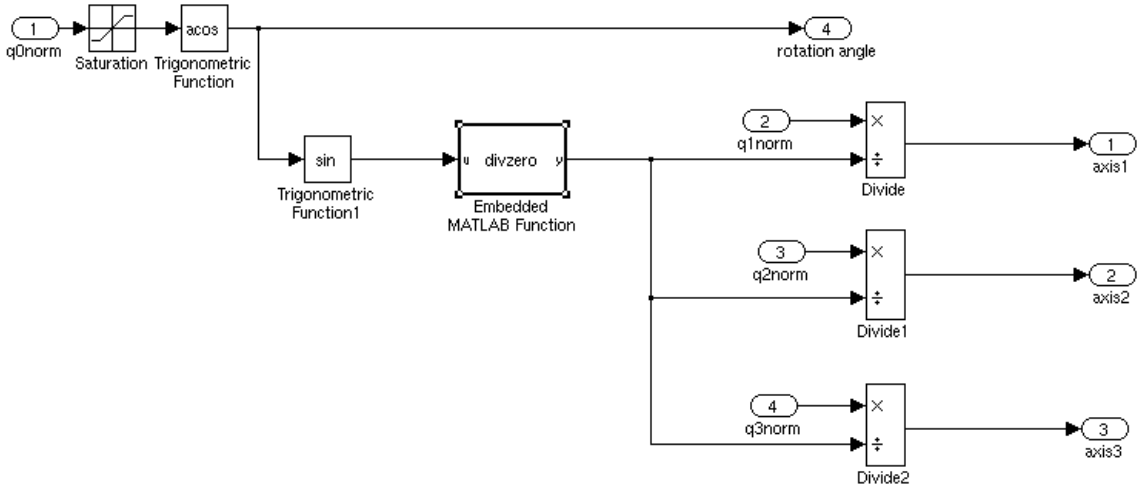

Figure C.21: Quaternion to Angle Axis Subsystem

This is the gyroshift.m function found in the visual level of the Attitude Determination Model:

```
function y = gyroshift(u)
% This block recenters the gyro euler angles to -pi and pi.

while u(3)>pi/2
    u(3)=u(3)-pi;
end

while u(3)<-pi/2
    u(3)=u(3)+pi;
end
while u(1)>pi
    u(1)=u(1)-pi;
end

while u(1)<-pi
    u(1)=u(1)+pi;
end

while u(2)>pi
    u(2)=u(2)-pi;
end
```

This is a function that prevents a discontinuity in the transformation from quaternions to the angle axis notation for the virtual reality block:

```
function y = divzero(u)
% This block prevents dividing by zero
if u == 0
    y = 1;
else
    y = u;
end
```

This is a function that transforms a quaternion to a 3-2-1 Euler Angle Sequence in the Attitude Determination - Top Level block:

```
function [psi,theta,phi] = Quat2Eul2(q0,q1,q2,q3)
% Produces Euler angle Array [Psi Theta Phi]'
% (NASA 321 rotation) from Quaternion

m11 = 2*q0^2+2*q1^2-1;
m12 = 2*q1*q2+2*q0*q3;
m13 = 2*q1*q3-2*q0*q2;
m23 = 2*q2*q3+2*q0*q1;
m33 = 2*q0^2+2*q3^2-1;

psi = atan(m12/m11);
theta = asin(-m13);
phi = atan(m23/m33);
```

This function takes the pqr and transforms it into a quaternion rate in the PQR to quatrernion dot block.

```
function [qdot0,qdot1,qdot2,qdot3] = b2qdot(P,Q,R,q0,q1,q2,q3)
%description-roll,pitch,yaw to qdot

quatdot = .5*[0 -P -Q -R; P 0 R -Q; Q -R 0 P; R Q -P 0]*
[q0;q1;q2;q3];
qdot0 = quatdot(1);
qdot1 = quatdot(2);
qdot2 = quatdot(3);
qdot3 = quatdot(4);
```

Below is the VRML code for the visual SimSat models used:

```
#VRML V2.0 utf8

WorldInfo {
  title "VRML Satellite"
  info ["Copyright 2004 Jason Smith"
   "$Revision: .2 $"
   "$Date: 2004/08/02 03:08:43 $"
   "$Author: Jason Smith $" ]
}
NavigationInfo {
```

```
  type "EXAMINE"
  headlight FALSE
}
Background {
  skyColor 0 0 0.4
}
Transform {
  translation 9 -2 0
  rotation 0 0 1 1.75
  children Billboard {
      children Shape {
          geometry Text {
              length 0
              fontStyle FontStyle {
                  topToBottom TRUE
                  style "PLAIN"
                  spacing 1
                  size 2
                  leftToRight TRUE
                  justify "BEGIN"
                  horizontal TRUE
                  family "SANS"
                }
              string "green == simulated SIMSAT"

            }
        }
    }
}
DEF Membrane Transform {
  rotation 0.00910466 -0.0928582 -0.995638 1.56053
  children [
    Transform {
      translation 0 -0.00263703 0
      scale 1 0.9974 1
      children Shape {
          geometry Sphere {
              radius 1
            }
          appearance Appearance {
              material Material {
                    diffuseColor .4 .9 .2
```

```
                shininess .7
            }
        }
    }
}
Transform {
  translation 5 0 0
  children Shape {
      geometry Box {
          size 3 3 3
        }
      appearance Appearance {
          material Material {
                diffuseColor .4 .9 .2
                shininess .7
            }
        }
    }
}
Transform {
  translation -5 0 0
  children Shape {
      geometry Box {
          size 3 3 3
        }
      appearance Appearance {
          material Material {
                diffuseColor .4 .9 .2
                shininess .7
            }
        }
    }
}
Transform {
  translation -2.5 0 0
  rotation 0 0 1 1.5708
  children Shape {
      geometry Cylinder {
          radius 0.25
          height 3.62623
        }
      appearance Appearance {
```

```
              material Material {
                    diffuseColor .4 .9 .2
                    shininess .7
                }
            }
        }
    }
    Transform {
      translation 2.5 0 0
      rotation 0 0 1 1.5708
      children Shape {
          geometry Cylinder {
              radius 0.25
              height 3.88468
            }
          appearance Appearance {
              material Material {
                       diffuseColor .4 .9 .2
                    shininess .7
                }
            }
        }
    }
    ]

}

DEF VIRTUAL Transform {
  rotation 0.00910466 -0.0928582 -0.995638 1.56053
  children [
    Transform {
      translation 0 -0.00263703 0
      scale 1 0.9974 1
      children Shape {
          geometry Sphere {
              radius 1
            }
          appearance Appearance {
              material Material {
                }
            }
        }
```

```
}
Transform {
  translation 5 0 0
  children Shape {
      geometry Box {
          size 3 3 3
        }
      appearance Appearance {
          material Material {
            }
        }
    }
}
Transform {
  translation -5 0 0
  children Shape {
      geometry Box {
          size 3 3 3
        }
      appearance Appearance {
          material Material {
            }
        }
    }
}
Transform {
  translation -2.5 0 0
  rotation 0 0 1 1.5708
  children Shape {
      geometry Cylinder {
          radius 0.25
          height 3.62623
        }
      appearance Appearance {
          material Material {
            }
        }
    }
}
Transform {
  translation 2.5 0 0
  rotation 0 0 1 1.5708
```

```
      children Shape {
          geometry Cylinder {
              radius 0.25
              height 3.88468
            }
          appearance Appearance {
              material Material {
                }
            }
        }
    }
    ]

}
DEF View1 Viewpoint {
  description "Original view"
  position 0 1 0
  fieldOfView 0.25
}
DEF View2 Viewpoint {
  description "View along Z"
  position 0 0 75
  fieldOfView 0.25
}
DEF DirLight DirectionalLight {
  direction 0.5 1 0.4
  color 1 0.5 0
  ambientIntensity 1
}
DEF PointLight PointLight {
  radius 100
  location 40 100 20
  color 0 0.7 0.7
  attenuation 1 0 0
  ambientIntensity 1
}
DEF dat3 Viewpoint {
  description "dat3"
  position -11.4174 1.73566 73.1001
  orientation -0.0813016 -0.0438259 0.995726 1.78818
  fieldOfView 0.19502
}
```

This is the PD_Dual_Sim.mdl Initialization file, pd_dualsim.m:

```
clc
close all
% clear all
global I11 I22 I33 Iw N1 N2 N3 d1 d2 d3 satnow
% ***********TARGET ATTITUDE*****************
th_target=60
% ***************************************************
% *                                                 *
% *                CONTROL VARIABLES                *
% *                                                 *
% ***************************************************
ep_K_T=.1    %thruster K "do nothing" limit

K_r_T=13.73  %thruster controller's rate gain

K_d_T=1      %thruster controller's delta gain
K_o_T=1      %thruster controller's overall gain

W_scale=550

K_r_W=K_r_T*W_scale   %wheel controller's rate gain
K_d_W=K_d_T*W_scale   %wheel controller's delta gain
K_o_W=-1.1     %wheel controller's overall gain

eswitch=.1    %control switching threshhold

% ***************************************************
% *                SYSTEM CONSTANTS                 *
% ***************************************************
% Wheel motor gains
K_motor_in=.1
K_motor=1

% acc_out=.5
% acc_in=6
%
% brake_out=1.5
% brake_in=1.5
```

```
T=.045          %positive thrust
T_bias=1    %negative to positive thrust ratio

T_av=T*(1+T_bias)/2

% Wheel MOI
Iw=66.17/32.171


% SIMSAT MOIs
I11=3800.66/32.2
I22=38318/32.2
I33=36652/32.2

% ***************************************************
% time
dt=.05       % time step
tend=120   % end time

satnow=[0 0 0]
th_1=0
th_2=0
th_3=0

w_1=0
w_2=0
w_3=0

%Max wheel speed
Om_max=3400*2*pi/60

%Max wheel torque
Tq_max=760/16

% Thruster moment arms (in)
d1=12
d2=36
d3=36

% Number of thrusters per axis
N1=1
N2=1        % Internal Sum adds to 3 (4 Feb)
```

```
N3=1

%Energy Poly Coef
P2_P=10
P2_K=100

%Braking constant
K_brake=1

% Voltage to send to D-Space (volts)
V_on=1

% D-space relay on/off settings
R_on=1
R_off=0
```

Plotting quaternion and euler-angle graphs from test runs, modelgraphs.m:

```
% This Matlab script plots the euler-angles and quaternions
% from the last simulation run

figure(1)
subplot(1,2,1)
plot(quaternions.time,quaternions.signals.values)
xlabel('time (s)')
title('Quaternions')
legend('q0 Gyro','q1 Gyro','q2 Gyro','q3 Gyro',
'q0 Model','q1 Model','q2 Model','q3 Model')
subplot(1,2,2)
plot(Euler.time,Euler.signals.values)
xlabel('time (s)')
ylabel('Angle (\circ)')
title('Euler Angles')
legend('\psi Gyro','\theta Gyro','\phi Gyro',
'\psi Model','\theta Model','\phi Model')

figure(2)
plot(Euler.time,Euler.signals.values)
xlabel('time (s)')
ylabel('Angle (\circ)')
title('Euler Angles')
legend('\psi Gyro','\theta Gyro','\phi Gyro',
```

```
'\psi Model','\theta Model','\phi Model')
```

This is the initialization file for the Simsat Experiment, initialize.m

```
% Wheel MOI
Iw=2.08

% SIMSAT MOIs
I11=9.141779699849999e+01
I22=9.576071189000000e+02
I33=9.606816110500000e+02

dt=.05;

tend = 30
```

This is the script, data.m, used to create the telemtry signal from the .mat file captured from SimSat telemtry data

```
% This script takes SimSat telemetry data from a .mat file.
% The telemetry data is assumed that the data captured is
% in the following order 'angle3';'angle2';'angle1';
% 'anglerate3';'anglerate2'; 'anglerate1';'wheelrate3';
% 'wheelrate2'; 'wheelrate1';
%
% Load the .mat file into the workspace and insert the
% file name in the next line.

dataname = yaw6 % yaw6 is the file name from file yaw6.mat

t = dataname.X.Data';
x1 = dataname.Y(1,1).Data';
y1 = dataname.Y(1,2).Data';
z1 = dataname.Y(1,3).Data';
x2 = dataname.Y(1,4).Data';
y2 = dataname.Y(1,5).Data';
z2 = dataname.Y(1,6).Data';
x3 = dataname.Y(1,7).Data';
y3 = dataname.Y(1,8).Data';
z3 = dataname.Y(1,9).Data';
```

```
%thrust
[m,n] = size(x1);
zerothrust = zeros(m,n);

c{1,1} = x1;
c{2,1} = y1;
c{3,1} = z1;
c{4,1} = x2;
c{5,1} = y2;
c{6,1} = z2;
c{7,1} = x3;
c{8,1} = y3;
c{9,1} = z3;
c{10,1} = zerothrust;
c{11,1} = zerothrust;
c{12,1} = zerothrust;

siglabels{1,1} = 'angle3';
siglabels{1,2} = 'angle2';
siglabels{1,3} = 'angle1';
siglabels{1,4} = 'anglerate3';
siglabels{1,5} = 'anglerate2';
siglabels{1,6} = 'anglerate1';
siglabels{1,7} = 'wheelrate3';
siglabels{1,8} = 'wheelrate2';
siglabels{1,9} = 'wheelrate1';
siglabels{1,10} = 'thrust1';
siglabels{1,11} = 'thrust2';
siglabels{1,12} = 'thrust3';

block = signalbuilder([],'create', t, c,siglabels);
```

Bibliography

1. Baruh, Haim. *Analytical Dynamics*. McGraw-Hill, Inc., 1999.
2. Colebank, James E., et al. *SIMSAT: A Satellite System Simulator and Experimental Test Bed for Air Force Research*. MS thesis, Air Force Institute of Technology (AETC), March 1999.
3. Dabrowski, Vincent J. *Experimental Demonstration Of An Algorithm To Detect The Presence Of A Parasitic Satellite*. MS thesis, Air Force Institute of Technology (AETC), March 2003.
4. Deb, S., et al. "Remote diagnosis of the International Space Station Utilizing Telemetry Data," *Proceedings of the SPIE - The International Society for Optical Engineering*, *4389*:60–71 (April 2001).
5. Dyne, S., et al. "Satellite Mechanical Health Monitoring," *IEEE Colloquium on 'Advanced Vibration Measurements, Techniques and Instrumentation for the Early Prediction of Failure'*, (105):4/1–8 (May 1998).
6. French, David B. *Hybrid Control Strategies for Rapid, Large Angle Satellite Slew Maneuvers*. MS thesis, Air Force Institute of Technology (AETC), March 2003.
7. Fulton, Joseph M. *Attitude Control and Multimedia Representaion of Air Force Institute of Technolgy's (AFIT's) Simulation Satellite (SimSat)*. MS thesis, Air Force Institute of Technology (AETC), March 2000.
8. Goddard Space Flight Center. "Attitude Determination and Control," *http://mabwww.gsfc.nasa.gov/products/attitude/* (2003).
9. Johnson, R. W. and S. Jayaram. "A New Real-Time Automated Ground Health Monitoring System at a Satellite Ground Control Station," *International Journal of Control and Intelligent Systems*, *32*(1):27–34 (2004).
10. Kim, ByungMoon, et al. *A SPACECRAFT SIMULATOR FOR RESEARCH AND EDUCATION*. MS thesis, Georgia Institute of Technology, 2002.
11. Kimsal, Matthew B. *Design of a Space-Born Autonomous Infrared Tracking System*. MS thesis, Air Force Institute of Technology (AETC), March 2004.
12. Kuipers, Jack B. *Quaternions and Rotation Sequences*. Princeton University Press, 2002.
13. Morris, Conrad and Derek Rothwell. "Operational Spacecraft Simulations: Present and Future," *Simulation and Modelling of Satellite Systems, 2002. IEE Seminar and Exhibition (Ref. No. 2002/074)* (April 2002).
14. NASA. "Mars Rover Takes Baby Steps," *http://www.jpl.nasa.gov/solar_system/features/rovers_111202.cfm* (November 2002).

15. Northrop Grumman Navigation Space Sensors Division. "LN-200 Product Description," *http://www.nsd.es.northropgrumman.com/Html/LN-200/* (2005).

16. SkEyes Unlimited Corporation. "sk_imu_manual." 2003.

17. Wertz, Larson. *Space Mission Analysis and Design* (3 Edition). Space Technology Library, 1999.

18. Wiesel, William E. *Spaceflight Dynamics* (2 Edition). McGraw-Hill, Inc., 1997.

Vita

Capt Jason Smith graduated from the United States Air Force Academy in 2000 with a BS in Aeronautical Engineering. Following graduation, in May of 2000, he was stationed at Wright Patterson AFB, OH where he was assigned as a F118-GE-100 engine engineer, the engine currently flying on the B-2 aircaft. Following AFIT, Capt Smith will be assigned to the Space Vehicles Directorate of the Air Force Research Laboratory at Kirtland AFB, NM.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
21 Mar 05	**Master's Thesis**	15 Aug 03 – 21 Mar 05

4. TITLE AND SUBTITLE

ATTITUDE MODEL OF A REACTION WHEEL/ FIXED THRUSTER BASED SATELLITE USING TELEMETRY DATA

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Smith, Jason, E., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENY/05-M010

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Withheld

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Attitude determination of satellites is normally the job of inertial instruments, such as gyroscopes, or through sensing instruments, such as star trackers or Global Positioning Satellites (GPS). Satellite health monitoring systems watch and determine if the satellite deviates from its normal operating attitude orientation. Knowing the orientation of a satellite is essential in being able to control it in order to complete the satellite's designated mission. While there are a multitude of ways to determine a satellite's orientation, very little research has been done on determining if the attitude of a satellite can be determined directly from telemetry data of the attitude control systems and an accurate spacecraft model. The fidelity of a satellite attitude determination model required to get reasonable predictions from using only telemetry data of the attitude controllers, such as thruster on/off indicators and reaction wheel rotor speeds, is investigated. Experimental tests using telemetry data received from the Air Force Institute of Technology's (AFIT) Simulated Satellite, SimSat, is used in verifying a Matlab model which outputs SimSat's orientation from SimSat's reaction wheel and thruster telemetry data. Software modeling results showed that it is possible to determine a satellite's attitude from only the attitude controllers' telemetry data when the satellite's dynamic model is known. Testing involving SimSat showed that attitude determination from the Matlab model is possible but not perfect. Additional information needs to be known about the satellite's systems and characteristics and about the environment in which the satellite operates, in order to increase the fidelity of the model for more accurate predictions of the satellite's attitude. Even though more research is needed, there is promise for using satellite attitude controllers for attitude determination in fields such as health monitoring and modeling and simulations.

15. SUBJECT TERMS

Attitude Control; Attitude Determination; Momentum Wheels; Reaction Wheels; Thrusters; Rigid Body Dynamics; Rotational Kinematics; Angular Momentum; Satellite Simulator; Hardware Simulation; Satellite Dynamics; Telemetry Data

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr. Richard Cobb
U	**U**	**U**	**UU**	137	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4559; e-mail: richard.cobb@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18